UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

BIBLIOGRAPHY

AGENTIS, n.d. *Simplifying the complexity of application development. Developing and deploying J2EE solutions with Agentis Adaptive Enterprise TM Solution Suite.* Atlanta, GA: Agentis International, Inc.

AGERFALK, P.J. 2004. Grounding through operationalization. Constructing tangible theory in IS Research. *Proceedings of the 12th European Conference on IS (ECIS 2004), Turku, Finland, 14-16 June.*

AKHLAGHPOUR, S., WU, J., LAPOINTE, L. & PINSONNEAULT, A. 2009. Re-examining the status of "IT" in IT research - An update on Orlikowski and Iacono (2001). *Americas Conference on Information Systems (AMCIS) Proceedings, San Francisco, California, August 6-9, pp. 1-14.*

ALEXANDER, P.M. 2002. *Towards reconstructing meaning when text is communicated electronically*. Pretoria: University of Pretoria (PhD Thesis, University of Pretoria). [Online]. Available: http://upetd.up.ac.za/thesis/available/etd-08192002-155431/ [Cited 10 May 2006].

AVISON, D. & FITZGERALD, G. 2006. *Information systems development*: *Methodologies, techniques and tools*. Maidenhead, UK: McGraw Hill.

AVISON, D.E. & FITZGERALD, G. 1995. *Information systems development: Methodologies, techniques and tools*. 2nd ed. Maidenhead: McGraw Hill.

AVISON, D.E., DWIVEDI, Y.K., FITZGERALD, G. & DOWELL, P. 2008. The beginnings of a new era: Time to reflect on 17 years of the ISJ. *Information Systems Journal*, vol. 18, no. 1, pp. 5-21.

AβMANN, U., ZSCHALER, S. & WAGNER, G. 2006. Ontologies. Meta-models and the model driven paradigm. In *Ontologies for Software Engineering and Software Technology* (Coral Calero, Fransisco Ruiz and Mario Piattini, eds.). Berlin: Springer Verlag, pp. 249-274.

BAKER, R. 2006. *What software crisis?* [Online]. Available: tech.norable.com/2006/05/what-software-crisis.html [Cited 3 November 2008].

BASDEN, A. 2001. Christian philosophy and Information Systems. *Presented to Institute for Christian studies, Toronto.* [Online]. Available: http://www.isi.salford.ac.uk/dooy/papers/cpis.html [Cited on 31 August 2006].

BENSON, S. & STANDING, C. 2005. *Information Systems: A business approach*, 2nd ed. Australia: Wiley & Sons.

BENSTA, H. 2010. Conceptualization of a framework based on ontology for construction an object model. *Proceedings of the IBIMA 2010 conference: Business transformation through innovation and knowledge management: An academic perspective, Istanbul, Turkey,* June 23-24, pp. 2058-2070.

BERG, B.C. 2007. *Qualitative research methods for the social sciences.* 6th ed. Boston, MA: Pearson Education Inc.

BEYNON, M., BOYATE, R. & CHAN, Z.E. 2008. Intuition in software development revisited. *Proceedings of the 20th Annual Psychology of Programming Interest Group Conference, Lancaster University, UK, September 10-12.*

BHASKAR, R. 1989. *The possibility of naturalism:* A philosophical critique of the contemporary human sciences, 2nd ed. Hemel Hempstead: Harvester Wheatsheaf.

BJØRNER, D. 2008. *Software engineering: An unended quest.* Denmark: Technology University of Denmark (Doctor of Technology Thesis, Technology University of Denmark).

BROOKS, F.P. 1987. *No silver bullet. Essence and accidents of software engineering.* Computer Magazine. [Online]. Available: http://virtualschool.edu/mon/softwareEngineering/BrooksNoSilverBullet.html [Cited 10 June 2008].

BROOKS, F.P. 1995. *The mythical man-month.* Anniversary ed. New York, NY: Addison-Wesley.

BROWN, I. 2008. Investigating the impact of the external environment on strategic information systems planning: A qualitative inquiry. *Proceedings of the 2008 annual research conference of the South African Institute of Computer Scientists and Information Technologists on IT research in developing countries: Riding the wave of technology*, *Garden*

Route, Wilderness, South Africa, October 6-8, ACM International Conference Proceeding Series, vol. 338, pp. 8-15.

BROWN, R., NERUR, S. & SLINKMAN, C. 2004. The philosophical shifts in software development. *Proceedings of the Tenth Americas Conference on Information Systems, New York, August 5-8.*

BROWN, A.W., WALLNAU, K.C. & FEILER, P.H. 1991. Understanding integration in a software development environment: Issues and illustrations. *Journal of Systems Integration,* vol. 3, no. 3-4, pp. 303-329.

BRYL, V., GIORGINI, P. & MYLOPOULOS, J. 2009. Designing socio-technical systems: From stakeholder goals to social networks. *Requirements Engineering Journal,* vol. 14, no. 1 pp. 47-70.

BRYMAN, A. 2004. *Social research methods.* 2nd ed. Oxford: Oxford University Press.

BUITELAAR, P., CIMIANO, P. & MAGNINI, B. 2003. Ontology learning from the text: An Overview. In *Ontology Learning from Text: Methods, Applications and Evaluation*. Edited by Buitelaar, P., Cimiano, P. and Magnini B.

BURRELL, G. & MORGAN, G. 1979. *Sociological paradigms and organisational analysis.* London: Heinemann.

CALERO, C., RUIZ, F. & PIATTINI, M. 2006. *Ontologies for software engineering and software technology*. London: Springer Verlag.

CAMPBELL, B. 2009. A resolution of student's grounded theory a priori reading dilemma. *Proceedings of Conf-IRM, AI-Ain, UAE, May 21-23.*

CARNEGIE MELON SOFTWARE ENGINEERING INSTITUTE. n.d. *Software product lines*. [Online]. Available: http://www.sei.cmu.edu/productlines/ [Cited 14 August 2006].

CHARMAZ, K. 2000. Grounded Theory: Objectivist and constructivist methods. In *Norman K. Denzin & Yvonna S. Lincoln (eds.), Handbook of Qualitative Research,* 2nd ed., pp. 509-535. Thousand Oaks, CA: Sage.

CHARMAZ, K. 2006. *Constructing Grounded Theory: A practical guide through qualitative analysis.* Thousand Oaks, CA**:** Sage.

CHECKLAND, P. 1999. *Systems thinking, systems practice: Includes a 30-year retrospective.* Chichester: Wiley.

CORCHO, O., FERNANDEZ-LOPEZ, M. & GOMEZ-PEREZ, M. 2006. Ontological engineering: Principles, methods, tools and languages. In *Ontologies for Software Engineering and Software Technology*. Edited by Coral Calero, Fransisco Ruiz and Mario Piattini. Berlin: Springer Verlag.

CORDEIRO, J. & FILIPE, J. n.d. *Language action perspective. Organizational semantics and the theory of organized activity. A comparison.* [Online]. Available: http://ltodi.est.ips.pt/jcordeiro/documents/demo2003CordFil_Final.pdf [Cited 4 May 2008].

CORNFORD, T. & SMITHSON, S. 1996. *Project research in Information Systems: A student guide.* New York, NY: Palgrave.

CRETU, L.G. 2010. Business process oriented software engineering. *Proceedings of the IBIMA 2010 conference: Business transformation through innovation and knowledge management: An academic perspective, Instanbul, Turkey, 23-24 June 2010*, pp. 1773-1786.

DAHLBOM, B. & MATHIASSEN, L. 1997. The future of our profession. *Communications of the ACM*, vol. 40, no. 6, pp. 499-579.

DAHLBOM, B. & MATHIASSEN, L. 1992. *Systems development philosophy. Computers and society.* ACM SIGCAS Computers and Society, vol. 22, no. 1-4, 1992.

DAHLBOM, B. & MATHIASSEN, L. 1993. *Computers in context. The philosophy and practice of systems design.* Oxford: NCC Blackwell.

DAHLBOM, B. 1996. The new Informatics. *Scandinavian Journal of Information Systems*, vol. 8, no. 2, pp. 29-48.

DE OLIVEIRA, K.M., VILLELA, K., ROCHA, A.R. & TRAVASSOS, G.H. 2006. Use of ontologies in software development environments. In *Ontologies for software engineering and software technology*. Edited by Coral Calero, Fransisco Ruiz and Mario Piattini (eds.). Berlin: Springer Verlag.

DENNIS, A., WIXON, B.H. & TEGARDEN, D. 2002. *Systems analysis and design: An object oriented approach with UML*. New York, NY: Wiley.

DICTIONARY.COM. n.d. *Theory*. [Online]. Available: http://dictionary.reference.com/browse/theory/ [Cited 27 June 2010].

DILLEY, P. 1999. Queer theory: Under construction. *QSE: International Journal of Qualitative Studies in Education*, vol. 12, no. 5, pp. 457-472.

DING, Y. & FOO, F. 2002. Ontology research and development. Part 1- a review of ontology generation. *Journal of Information Science*, vol. 28, no. 2, pp. 123-136.

DITTRICH, Y. & SESTOFT, P. 2005. *Designing evolvable software products: Coordinating the evolution of different layers in kernel based software products.* [Online]. Available: http://www.itu.dk/research/sdg/doku.php [Cited October 2006].

DOBING, B. & PARSONS, J. 2007. What practitioners are saying about the Unified Modelling Language. In *Managing Worldwide Operations and Communications with Information Technology , Information Resources Management Association International Conference, Vancouver, British Columbia, Canada, May 19-23*, pp. 123-126.

DRISTAS, S., GYMNOPOULOS, L., KARYDA, M., BALOPOULOS, T., KOKOLAKIS, S., LAMBRINOUDAKIS, C. & GRITZALIS, S. 2005. Employing ontologies for the development of security critical applications: The secure e-poll paradigm. In *Challenges of Expanding Internet: E-Commerce, E-Business, and E-Government, IFIP International Federation for Information Processing*, vol. 189, pp. 187-20.

DU PLOOY, N.F. 2004. *Information technology change management.* Pretoria:University of Pretoria (class notes), South Africa, 2004.

EL BAZE, N. 2005. Cracking software development complexity: In order to reap the benefits of progress in processor technology, we must fundamentally rethink how to write software. *Line 56-The Business Executive daily*. [Online], Available: http://www.partechvc.com/newspdfs/Line56_NEB.pdf [Cited 30 March 2005].

ENGINEERS COUNCIL FOR PROFESSIONAL DEVELOPMENT (ECPD). [Online] Available: http://www.abet.org/history [Cited on 11 January 2008].

FALBO, R.A., GUIZZARDI, G. & DUARTE, K.C. 2002. An ontological approach to domain engineering. *International Conference of Software Engineering and Knowledge Engineering, SEKE'02, Ischia, Italy, July 15-19.*

FENSEL, D. 2004. *Ontologies: A silver bullet for knowledge management and electronic commerce,* 2nd edition. Berlin: Springer Verlag.

FITZGERALD, B. & HOWCROFT, D., 1998. Competing dichotomies in IS research and possible strategies for resolution. *ICIS '98 Proceedings of the International Conference on Information systems (ICIS '98), Atlanta, GA: Association for Information Systems, Helsinki,* pp. 155-164.

FUENTES-FERNÁNDEZ, R. & GOMEZ- SANZ, J.J. 2009. Understanding the human context in requirements engineering. *Requirements Engineering Journal,* vol. 14, Springer-Verlag.

GARCIA-DUQUE, J., PAZOS-ARIAS, J.J., LOPEZ-NORES, M., BLANCO-FERNANDEZ, Y., FERNANDEZ-VILAS, A., DIAZ-REDONDO, R.P., RAMOS-CABRER, M. & GIL-SOLLA, A. 2009. Methodologies to evolve formal specifications through refinement and retrenchment in an analysis-revision cycle. *Requirements Engineering Journal* , vol. 14, no. 3, pp. 129-153.

GASCHE, R. 1986. Infrastructures and systematicity. In *Deconstruction and philosophy: The texts of Jacques Derrida,* edited by John Sallis, pp. 3-20.

GASSON, S. 2003. Rigor in grounded theory research: An interpretive perspective on generating theory from qualitative field studies. In *The Handbook of Information Systems Research,* edited by M.E. Whitman & A.B. Woszczynski. Hershey, PA: Idea Group Publishing.

GERBER, A.J. 2006. *Towards a comprehensive functional layered architecture for the semantic web*. Pretoria: UNISA (PhD thesis in Computer Science, UNISA).

GIDDENS, A. 1976. *New rules of sociological method: A positive critique of interpretative sociologies*. London: Hutchinson.

GILB, T. 1985. Evolutionary delivery versus the "waterfall model". *ACM SIGSOFT Software Engineering Notes,* July 1985, vol.10, no. 3, pp. 49-61, July 1985.

GLASER, B,G. & STRAUSS, A.L. 1965. *Awareness of dying,* Chicago, IL: Aldine.

GLASER, B.G. & STRAUSS, A.L. 1967. *The discovery of Grounded Theory: Strategies for qualitative research.* New York, NY: Aldine De Gruyter.

GLASER, B.G. 1978. *Theoretical sensitivity: Advances in the methodology of grounded theory.* Mill Valley, CA: Sociology Press.

GLASER, B.G. 1998. *Doing grounded theory: Issues and discussions*. Mill Valley, CA: Sociology Press.

GLASER, B.G. 1992. *Basics of Grounded Theory analysis: Emergence vs. forcing.* Mill Valley, CA: Sociology Press.

GLASER, B.G. 1993. *Examples of Grounded Theory: A reader*. Mill Valley, CA: Sociology Press.

GLASER, B.G. 2002. Constructivist Grounded Theory? In *Forum Qualitative Sozialforschung / Forum: Qualitative Social Research,* vol. 3, no. 3. [Online]. Available: http://www.qualitative-research.net/fqs/fqs-eng.htm [Cited 15 March 2008].

GOEDE, R. 2005. *Framework for the explicit use of specific systems thinking methodologies in data driven decision support system development.* Pretoria: University of Pretoria (PhD Thesis, University of Pretoria). [Online]. Available: http://upetd.up.ac.za/thesis/available/etd-05132005-080727/ [Cited 12 January 2008].

GOEDE, R. *Substantive area of research.* Personal e-mail (23 April 2008).

GOLDKUHL, G. 1999. The grounding of usable knowledge: An inquiry in the epistemology of action knowledge. In *Högskolor och Samhälle i Samverkan (HSS99), Linköping University,* March 16-18 (CMTO Research Papers 1999:03).

GOLDKUHL, G. 2002. Anchoring scientific abstractions: Ontological and linguistic determination following socio-instrumental pragmatism. *Proceedings of the European*

Conference on Research Methods in Business and Management Studies (ECRM 2002), MCIL, Reading, UK, April 29-39.

GOMES, P. 2004. Software design retrieval using Bayesian networks and WordNet. *Proceedings of the 7th European Conference on Case Based Reasoning (ECCBR, 2004), Madrid, Spain, August 30 - September 2*, pp. 184-197. [Online]. Available: http:// eden.dei.uc.pt/~pgomes/ [Cited 14 August 2006].

GONZALEZ-PEREZ, C. & HENDERSON-SELLERS, B. 2006. An ontology for software development endeavors. In *Ontologies for Software Engineering and Software Technology.* Edited by Coral Calero, Fransisco Ruiz and Mario Piattini. Berlin: Springer Verlag.

GREEN, R.M. 1998. Heuristic power as the test of theory: A response to Franscisca Cho. *The Journal of Religious Ethics,* vol. 26, no. 1 , pp. 175-184.

GREGOR, S. 2006. The nature of theory in Information Systems. *MIS Quarterly*, vol. 30, no. 3, pp. 612-642.

GREGOR, S. n.d. *The struggle towards an understanding of theory in Information Systems.* School of Business and Information Management, Australian National University. [Online]. Available: http://epress.anu.edu.au/info_systems/ part-ch01.pdf [Cited 31 August 2006].

GRUBER, T. 2008. Ontology. In *Encyclopedia of Database Systems, edited by Ling Liu and M. Tamer Ozsu.* Berlin: Springer Verlag.

GRUBER, T.A. 1993. Transition approach to portable ontology specifications. *Knowledge Acquisition*, vol. 5, no. 2, pp.199-220.

GUARINO, N. 1998. Formal Ontology and Information Systems. *Proceedings of FOIS'98, Trento, Italy, June 6-8*, 1998, pp. 3-15. Amsterdam: IOS Press.

GUARINO, N., CARRARA, M. & GIARETTA, P. 1994. Formalising ontological commitments. *Proceedings of the 12th National Conference on Artificial Intelligence (AAAI-94), Seattle, WA, August 1-4.*

GUNTER, C.A., GUNTER, E.L., HILL, M.N.J. & ZAVE, P. 2000. Formal software engineering. *ACM SIGSOFT Software Engineering Notes*, January 2000, vol. 25, no. 1, pp. 54.

GUZZARDI, G. & HALPIN, T. 2008. Ontological foundations for conceptual modeling, Guest Editorial, *Applied Ontology*, vol. 3, no 1-2, pp. 1-12.

HACKING, I. 2002. *Historical Ontology*. London: Harvard University Press.

HALLER, A. & OREN, E. 2006. *A process ontology to represent semantics of different process and choreography meta models*. Gallway: National University of Ireland (DERI Technical Report).

HARRIS, M.A. & WEISTROFFER, H.R. 2009. A new look at the relationship between user involvement in systems development and system success. *Communications of the Association of Information Systems,* vol. 24, no. 42, pp. 739-756.

HARRIS, M.L., HEVNER, A.R., & COLLINS, R.W. 2009. Controls in flexible software development. *Communications of the Association of Information Systems*, vol. 24, no. 43, pp. 757-776.

HEINEMAN, G.T. 2000. A model for designing adaptable software components. *ACM SIGSOFT, Software Engineering Notes,* vol. 25, no. 1, pp. 55-56.

HENNING, E. 2007. *Finding your way in qualitative research.* Pretoria: Van Schaik Publishers.

HEVNER, A.R., MARCH, S.T., PARK, J. & RAM, S. 2004. Design Science in Information Systems research. *MIS Quarterly*, vol. 28, no. 1, pp. 75-105.

HIGHSMITH, J. 2004. *Agile project management: Creating innovative products*. Boston: Addison-Wesley.

HIRSCHHEIM, R., KLEIN, H.K. & LYYTINEN, K. 1995. *Information systems development and data modelling: Conceptual and philosophical foundations*. Cambridge: Cambridge University Press.

HOFFERER, P. 2007. Achieving business process model interoperability using meta-models and ontologies. In *Managing Worldwide Operations and Communications with Information Technology, Proceedings of 2007 Information Resources Management Association, International Conference, Vancouver, British Columbia, Canada, May 19-23*, edited by M. Khosrow-Pour. Hershey, PA: IGI Publishing, pp. 1620-1631.

HOHMANN, C. 2007. Emotional digitization. A reflexive examination from the view of the industry. *International Journal of Technology and Human Interaction*, vol. 3, no. 1, pp. 17-29.

HOLT, A.W. 1997. *Organized activity and its support by computer.* Dordrecht: Kluwer Academic Publishers.

HUNTER, G.M. 2000. "Excellent" systems analysis: A grounded theory approach to qualitative research. In *The Handbook of Information Systems Research*, edited by Whitman, M.E. & Woszczynski, A.B. Hershey, PA: Idea Group Publishing.

HUNTER, G.M. 2003. Qualitative research in information systems: An exploration of methods. In *The Handbook of Information Systems Research,* edited by Whitman, M.E. & Woszczynski, A.B. Hershey, PA: Idea Group Publishing.

IEEE. 1990. *IEEE Standard Glossary of Software, IEEE Standard 610.12-1990.* New York, NY: IEEE.

IIVARI, J. 1991. A paradigmatic analysis of contemporary schools of IS development. *European Journal of Information Systems*, vol. 1, no. 4, pp. 249-272.

IIVARI, J., HIRSCHHEIM, R. & KLEIN, H.K. 1998. A paradigmatic analysis contrasting information systems development approaches and methodologies. *Journal of Information Systems Research*, vol. 9, no. 2, pp. 164-193.

IN-PHARMA TECHNOLOGIST.COM. 2005. *Improved data dealing drives drug discovery.* [Online]. Available: http://www.in-pharmatechnologist.com/news/ [Cited 30 March, 2005].

INTRONA, D. 1992. *Towards a theory of management information.* Pretoria: University of Pretoria (Unpublished DCom Dissertation).

INTRONA, L.D. 1997. *Management information and power: A narrative of the involved manager*. Bassingstoke: Macmillan Press.

ISABELLA, C.A. 1990. Evolving interpretations as a change unfolds: How managers construe key organizational events. *The Academy of Management Journal,* vol. 33, no. 1, pp. 7-41.

JACKSON, M.A. 1981. *A systems development method: Tools and notions for program construction*. Cambridge, UK: Cambridge University Press.

JAYARATNA, N. 1990. Systems analysis: The need for a better understanding. *International Journal of Information Management,* vol. 10, no. 8, pp. 228-234.

JOHN, R.R. 2003. When information come of age: Technologies of knowledge in the age of reason and revolution, 1700-1850. *William and Mary Quarterly*, vol. 60, no. 2, Reviews of Books.

JURISICA, I., MYLOPOULOS, J. & YU, E. 1999. Using ontologies for knowledge management: An information systems perspective. *In Proceedings of the 62nd Annual Meeting of the American Society for Information Science (ASIS99), Washington, DC, Oct 31-Nov 4*, pp. 482-496. Medford, NJ: Information Today, Inc.

KAPLAN, B. & MAXWELL, J.A. 1994. Qualitative research methods for evaluating computer information systems. In *Evaluating Health Care Information Systems: Methods and Applications* (J.G. Anderson, C.E. Aydin and S.J. Jay, eds.). Thousand Oaks, CA: Sage Publications, pp. 45-68.

KAWALEK, P. & LEONARD, J. 1996. Evolutionary software development to support organizational and business change: A case study account. *Journal of Information Technology,* vol. 11, no. 3, pp. 185-198.

KEEN, P.G.W. 1988. Roles and skills base for the IS organization. In *Elam, J.J., Ginzberg, M.J., Keen, P.G.W. & Zmud, R.W. 1988. Transforming the IS Organization.* Washington, DC: ICIT Press, pp. 17-40.

KIRLIDOG, M. & AYTOL, M. 2010. A longitudinal investigation about issues and problems of software engineering in Turkey. *Proceedings of the IBIMA 2010 Conference: Business Transformation through Innovation and Knowledge Management: An Academic Perspective, Istanbul, Turkey, June 23-24,* pp. 66-76.

KLEIN, H.K. & MYERS, M.D. 1999. A set of principles for conducting and evaluating interpretive field studies in Information Systems. *MIS Quarterly,* vol. 23, no. 1, pp. 67-94.

KROEZE, J.H. 2009. *Information Systems and the humanities: a symbiotic relationship?* Vanderbijlpark: North-West University, Vaal Triangle Campus. (Vaal Triangle Occasional Papers: Inaugural lecture 5/2009.)

KROEZE, J.H., LOTRIET, H., MAVETERA, N., POSTMA, D., PFAFF, M., SEWCHURRAN, K. & TOPI, H. 2010. Humanities-enriched Information Systems: Panel discussion (abstract). In *Proceedings of the 18th European Conference on Information Systems (ECIS), IT to Empower, University of Pretoria, June 6-10.*

KROEZE, J.H., LOTRIET, H.H., MAVETERA, N., PFAFF, M.S., POSTMA, D.J.V.R., SEWCHURRAN, K. & TOPI, H. 2011. ECIS 2010 panel report: Humanities-enriched Information Systems. *Communications of the Association for Information Systems Journal (CAIS),* vol. 28, no. 1. Available at: http://aisel.aisnet.org/cais/vol28/iss1/24 [Cited 27 October 2011].

KRUEGER, C.W. n.d. *Introduction to software product lines.* [Online]. Available: http://www.softwareproductlines.com/introduction/introduction.html [Cited 17 June 2008].

LATOUR, B. 1999. *Pandora's hope: Essays on the reality of science studies*. Cambridge, MA: Harvard University Press.

LEEDY, P.D. & ORMROD, J.E. 2005. *Practical research: Planning and design*. 8th ed. Upper Saddle River, NJ: Pearson Education International.

LEHMAN, M.M. 1991. Software engineering, the software process and their support. *Software Environments and Factories (special issue), IEEE Software Engineering Journal*, vol. 6, no. 5, pp. 243-258.

LEHMAN, M.M. 1994. Feedback in the software evolution process. Keynote address. In *CSR Eleventh Annual Workshop on Software Evolution: Models and Metrics.* Dublin, September 7-9, 1994. Also in: *Workshop Proceedings, Software Maintenance (special issue), Information and Software Technology, vol. 38, no. 11,* pp. 681-686.

LEHTOLA, L., KANPPANEN, M., VAHINIITTY, J. & KOMSSI, M. 2009. Linking business and requirements engineering: Is solution planning a missing activity in software product companies? *Requirements Engineering Journal,* vol. 14, no. 2, pp. 113-128.

LEMMENS, R. 2006. *Semantic interoperability of distributed geo-services*. Delft: TU Delft (PhD Dissertation, Delft University of Technology). [Online]. Available: http://www.ncg.knaw.nl/publicaties/geodesy/pdf/63lemmens.pdf [Cited 17 June 2008].

LIU, L. & YU, E. 2004. Designing information systems in social context: A goal and scenario modelling approach. *Journal of Information Systems,* vol. 29, no. 2, pp.187-203.

LOCKWOOD, D. 1964. Some remarks on ‘The Social System’. In Demereth, N.J. & Peterson, R.A. 1967. *System, change, and conflict: A reader on contemporary sociological theory and the debate over functionalism*. New York: Free Press.

LOFLIN, L. n.d. *Romanticism notes.* [Online]. Available: http://www.sullivan-county.com/nf0/nov_2000/romanticism.htm [Cited 4 September 2006].

MALINOWSKI, B. 1923. The problem of meaning in primitive languages. In *The Meaning of Meaning, edited by C.K. Ogden et al.* London: Routledge and Kegan Paul.

MALUF, D. & WIEDERHOLD, G. 1997. Abstraction of representation for interoperation. In *Foundation of Intelligent Systems, Lecture Notes in Computer Science, Lecture Notes in AI*, vol. 1525, pp. 441-455. Berlin: Springer.

MARCH, S.T. & SMITH, G.F. 1995. Design and natural science research on information technology. *Decision Support Systems*, vol. 15, no. 4, pp. 251-266.

MARTIN, P.Y. & TURNER, B.A. 1986. Grounded Theory and organizational tesearch. *The Journal of Applied Behavioral Science*, vol. 22, no. 2, pp. 141-157.

MATAVIRE, R. & BROWN, I. 2008. Investigating the use of ‘Grounded Theory’ in Information Systems research. In *SAICSIT '08: Proceedings of the 2008 annual research conference of the South African Institute of Computer Scientists and Information Technologists on IT research in developing countries: Riding the wave of technology, Wilderness, October 6-8*, pp. 139-147.

MAVETERA, N. & KADYAMATIMBA, A. 2003. A comprehensive agent mediated e-market framework. In *ACM International Conference Proceeding Series, Proceedings of the 5th international conference on electronic commerce, Pittsburgh, Pennsylvania*, vol. 50, pp. 158-164.

MAVETERA, N. & KROEZE, J.H. 2008. Practical issues in Grounded Theory Method research. *In the proceedings of the M & D Graduate Symposium, preceding SAICSIT 2008, Wildernis, October 6*.

MAVETERA, N. & KROEZE, J.H. 2009a. Issues affecting software development: A South African software practitioners' viewpoint. *Communications of the IBIMA*, vol. 9, no. 2. [Online]. Available: http://www.IBIMA.org [Cited 25 April 2009].

MAVETERA, N. & KROEZE, J.H. 2009b. Issues affecting software development: A South African software practitioners' viewpoint. *In Proceedings of the International Business Information Management Association Conference (IBIMA), Cairo, Egypt, January 4-6*.

MAVETERA, N. & KROEZE, J.H. 2009c. Practical considerations in Grounded Theory Method research. *Sprouts: Working Papers on Information Systems*, vol. 9, no. 32. [Online]. Available: http://sprouts.aisnet.org/9-32 [Cited 5 February 2010].

MAVETERA, N. & KROEZE, J.H. 2009d. A grounding framework for developing adaptive software products. In *Proceedings of the 13th IBIMA Conference on Knowledge Management and Innovation in Advancing Economies, Marrakech, Morocco, November 9-10*.

MAVETERA, N. & KROEZE, J.H. 2010a. Considerations in Grounded Theory research method: A reflection on the lessons learned. In *Proceedings of the 14th International Business Information Management Association Conference (14th IBIMA), Business Transformation through Innovation and Knowledge Management: An Academic Perspective, Istanbul, Turkey, June 23-24*, edited by Khalid S. Soliman, pp. 1454-1469.

MAVETERA, N. & KROEZE, J.H. 2010b. An ontology-driven software development framework. In *Proceedings of the 14th International Business Information Management Association Conference (14th IBIMA), Business Transformation Through Innovation and Knowledge Management: An Academic Perspective*, *Istanbul, Turkey, June 23-24,* edited by Khalid S. Soliman, pp. 1713-1724.

MAVETERA, N. & KROEZE, J.H. 2010c. Guiding principles for developing adaptive software products. *Communications of the IBIMA*, vol. 2010, Article ID 340296.

MAVETERA, N. 2004a. The use of ontologies in designing agents for e-markets: From a mechanist to a romantic approach. In *Proceedings of the International Business Information Management Conference (IBIMA '04), Amman, Jordan, July 4-6.*

MAVETERA, N. 2004b. The philosophy of ontologies: A new information systems development paradigm. In *Proceedings of the International Science and Technology Conference (ISTC'04), Vanderbijlpark, December 1-2.*

MAVETERA, N. 2007. A comprehensive ontology-driven software development architecture: A holistic approach to developing romantic software products. In *Managing Worldwide Operations and Communications with Information Technology, Proceedings of 2007 Information Resources Management Association, International Conference, Vancouver, British Columbia, Canada, May 19-23,* edited by M. Khosrow-Pour. Hershey, PA: IGI Publishing.

MAVETERA, N. 2010. Philosophical groundings in social science research. In *Fundamentals of Scientific Writing and Publishing: Research and Publication Series, Volume 1,* Mafikeng: Platinum Press.

MAVETERA, N. 2011. Towards an ontology-driven software development approach. In *Proceedings of the 16th IBIMA Conference on e-Government, Innovation and Knowledge Management: A Global Competitive Advantage, Kuala Lumpur, Malaysia, June 25-July* 2, pp. 1050-1057.

MERTON, R.K. 1967. *On Theoretical Sociology: Five essays, new and old.* Free Press, New York.

MESO, P. & JAIN, R. 2006. Agile software development: Adaptive systems principles and best practices. *Information Systems Management, vol. 23, no.* 3, June 2006, pp. 19-30.

MINGERS, J. 2002. Can social systems be autopoietic? Assessing Luhmann's social theory. *Sociological Review,* vol. 50, no. 2, pp. 278-299.

MOLENAAR, M. 1998. *An introduction to the theory of spatial object modelling for GIS.* London: Taylor & Francis.

MONOD, E. 2007. Editorial. Special issue on philosophy and epistemology: A Peter Pan Syndrome? *Information Systems Journal*, vol. 17, no. 2, pp. 133-141.

MULLET, D. 1999. *The software crisis*. [Online]. Available: http://www.unt.edu/benchmarks/archives/1999/july99/crisis.htm. [Cited 9 June 2008].

MYERS, M.D. 1997. Qualitative research in information systems. *MISQ Discovery,* archival version, vol. 21, no. 2, June 1997, pp. 241-242. [Online]. Available: http://www.misq.org/discovery/MISQD_isworld/ [Cited 9 April 2008].

MYERS, M.D. 2003. Qualitative research in Information Systems. *Association for Information Systems*. [Online] Available: http://www.qual.auckland.ac.nz/index.htm [Cited 9 April 2008].

NECHES, R., FIKES, R., FININ, T., GRUBER, T., PATIL, R., SENATOR, T. & SWARTOUT, W.R. 1991. Enabling technology for knowledge sharing. *AI Magazine*, vol. 12, no. 3, pp. 36-56.

NGWENYAMA, O.K. & LEE, A.S. 1997. Communication richness in electronic mail: Critical social theory and the contextuality of meaning. *MIS Quarterly*, vol. 21, no. 2, June, pp. 145-167.

OINAS-KUKKONEN, H. & HARJUMAA, M. 2009. Persuasive systems design: Key issues, process model, and system features. *Communications of the Association of Information Systems,* vol. 24, no. 28, pp. 485-500.

OLIVIER, S.M. 2004. *Information Technology research: A practical guide for Computer Science and Informatics.* 2nd ed. Pretoria: Van Schaik.

ORLIKOWSKI, W. J. & IACONO, S. 2001. Research commentary: Desperately seeking the “IT” in IT research: A call to theorizing the IT artefact. *Journal of Information Systems Research,* vol. 12, no. 2, pp. 121-134.

ORLIKOWSKI, W.J. 1993. CASE tools as organizational change: Investigating incremental and radical changes in systems development. *Management Information Systems Quarterly*, 340, vol. 17, no. 3, pp. 309-340.

PAGE, C. & MEYER, D. 2000. *Applied research design for business and management*. Australia: McGraw-Hill.

PEFFERS, K., TUUNANEN, T., ROTHENBERGER, M.A. & CHALTERJEE, S. 2008. A Design Science research methodology for Information Systems research. *Journal of Management Information Systems,* vol. 24, no. 3, pp. 45-77.

PRESSMAN, R.S. 2000. *Software engineering - A practitioner's approach.* 5th edition. Boston: Mc Graw-Hill Education.

PRESSMAN, R.S. 2005. *Software engineering - A practitioner's approach.* 6th edition. Boston: McGraw-Hill.

PURAO, S., BALDWIN, C., HEVNER, A., STOREY, V.C., PRIES-HEJE, J., SMITH, B. & ZHU, Y. 2008. The sciences of design: Observation on an emerging field. *Communications of the Association of Information Systems,* vol. 23, no. 28, pp. 523-546.

RANDELL, B. n.d. *The 1968 NATO software engineering reports.* [Online]. Available: http://homepages.cs.ncl.ac.uk/brian.randell/nato/natoreports/index.html [Cited 2 November 2008].

RAS, Z.W. & DARDZINSKY, A. 2004. Ontology based distributed autonomous knowledge systems. *International Journal of Information Systems*, vol. 29, no. 1, pp. 47-58.

REETLEY, A. 2003. *A literature review on grounded theory*. Johannesburg: Rand Afrikaans University (Masters Dissertation).

REPA, V. 2004. Methodology framework for information systems development. *CITSA Conference*, *Orlando, Florida, July 21-25*, pp. 2-5.

ROODE, J.D. 1993. *Implications for teaching of a process based research framework for Information Systems*. Pretoria: Department of Informatics, University of Pretoria.

ROODE, J.D. 2004. *Research methodologies and proposal (lecture)*. Pretoria: University of Pretoria (class notes).

ROQUE, L., ALMEIDA, A. & FIGUEIREDO, A.D. 2003. Context engineering: An IS development approach. *Proceedings: Action in Language, Organisations and Information*

Systems, International Conference (ALOIS'2003), Linköping, Sweden, March 12-13, pp. 107-122.

ROSENKRANZ, C. & HOLTEN, R. 2007. Towards measuring the complexity of information systems: A language critique approach. *Proceedings of International Resources Management Association (IRMA), Vancouver, British Columbia, May 19-23,* pp. 57-60.

RUIZ, F. & HILERA, J.R. 2006. Using ontologies in software engineering and technology. In *Ontologies for Software Engineering and Software Technology,* edited by Coral Calero, Fransisco Ruiz and Mario Piattini. Berlin: Springer Verlag.

SARANTAKOS, S. 1997. *Social research*. 2nd ed. New York, NY: Palgrave.

SCHACH, S.R. 2005. *Object oriented and classical software engineering*. 6th ed. New York, NY: McGraw-Hill.

SEP. 2008. *Phenomenology.* Stanford Encyclopaedia of Philosophy (SEP). [Online]. Available: http://palto.stanford.edu/entries/phenomenology/ [Cited 6 August 2009].

SHANKS, G. 1999. Semiotic approach to understanding representation in information systems. *Proceedings of the Information Systems Foundations Workshop on Ontology, Semiotics and Practice.* [Online]. Available: http://www.comp.mq.edu.au/isf99/shanks.htm [Cited 6 October 2005].

SHARMA, S. & INGLE, M. 2011. An ontology-driven information system. *International Journal of Computing Technology and Applications,* vol. 2, no. 1, pp. 147-154.

SHANKS, G., TANSLEY, E. & WEBER, R. 2003. Using ontology to validate conceptual models. *Communications of the ACM, vol. 46, no. 10, pp. 85-89.*

SHAWA, W. 2009. *An ontology for information systems development methodologies*. Potchefstroom: North West University (Master of Science Dissertation).

SHAWA, W., MAVETERA, N. & HUISMAN, M. 2009. Building language communities in organizational system development teams using ontologies. *Proceedings of the 13th IBIMA Conference on Knowledge Management and Innovation in Advancing Economies, Marrakech, Morocco, November 9-10*.

SIEGEMUND, K., THOMAS, E.J., ZHAO, U., PAN, J. & ASSMANN, U. 2011. Towards ontology-driven requirements engineering. *The 10th International Semantic Web Conference,* Bonn, October 23-27. [Online]. Available: http://iswc2011.semanticweb.org/fileadmin/iswc/papers/workshops/swese/4.pdf [Cited 18 October 2011].

SIMONOV, M., GANGEMI, A. & SOROLDONI, M. 2004. Ontology-driven natural language access to legacy and web services in the insurance domain. *Journal of Business Information Systems*, pp. 1-10.

SMITH, H. 2000. The role of ontological engineering in B2B net markets. *CSC Europe.* [Online]. Available: http://www.ontology.org/main/papers/csc-ont-eng.html [Cited 21 May, 2004].

SOFFER, P., GOLANY, B., DORI, D. & WAND, Y. 2001. Modelling off-the shelf information systems requirements: An ontological approach. *Requirements Engineering Journal*, vol. 6, no. 3, pp. 183-199.

SOWA, F.J. 2000. *Ontology, metadata and semiotics.* In *Conceptual structures in logical, linguistic and computational issues,* edited by Ganter and Mineau. Berlin: Springer-Verlag, pp. 55-81 (*Lecture notes in AI #1867).*

SOWA, F.J. 2006. A dynamic theory of ontology. In *Formal Ontology in Information Systems,* edited by B. Bennett & C. Fellbaum. Amsterdam: IOS Press, Amsterdam [Online]. Available: http://www.Jfsowa.com/pubs/dynonto.htm [Cited 9 June 2008].

SOWA, F.J. 1976. Conceptual graphs for a data base interface. *IBM Journal of Research and Development,* vol. 20, no. 4, pp. 336-357.

SOWA, F.J. n.d. *Building, sharing and merging ontologies*. [Online]. Available: http://www.JFsowa.com/ontology/ontoshar.htm [Cited on 16 August 2006].

STAMPER, R. 1992. Signs, organisations, norms and information systems. In *Proceedings of the 3rd Australian Conference on Information Systems*, *University of Wollongong, Australia, October 5-8.*

STOKES, D.R. & BIRD, J. 2008. Evolutionary robotics and creative constraints. In *Beyond the brain: Embodied, situated and distributed cognitions,* edited by Benoit Hardy-Vallée & Nicolas Payette. Newcastle: Cambridge Scholars Publishing, pp. 227-245.

STRAUSS, A.L. & CORBIN, J. 1990. *Basics of qualitative research: Grounded Theory procedures and techniques*. London: Sage.

STRUWIG, F.W. & STEAD, G.B. 2004. *Planning, designing and reporting research.* Cape Town: Pearson Eduaction.

STUDER, R., BENJAMINS, R. & FENSEL, D. 1998. Knowledge engineering: Principles and methods. *Data and Knowledge Engineering*, vol. 25, no. 1-2, pp. 161-197.

SUCHMAN, L.A. 1987. *Plans and situated actions: The problem of human-machine communications.* Cambridge: Cambridge University Press.

SUDDABY, R. 2006. From the editors: What Grounded Theory is not. *Academy of the Management Journal,* vol. 49, no. 4, pp. 633-642.

SUGUMARAN, V. & STOREY, V.C. 2002. Ontologies for conceptual modeling: Their creation, use and management. *Journal of Data and Knowledge Engineering*, vol. 42, no. 2, pp. 251-271.

SUGUMARAN, V. & STOREY, V.C. 2006. The role of domain ontologies in database design: An ontology management and conceptual modeling environment. *ACM Transactions on Database Systems (TODS),* vol. 31, no 3.

SUTTON, R.J. & STAW, B.M. 1995. What theory is not? *Administrative Science Quarterly*, ABI/Inform Global, September 1995, vol. 40, no. 3, pp. 371-384.

SWARTOUT, B., PATIL, R., KNIGHT, K. & RUSS, T. 1997. Toward distributed use of large scale ontologies. *Ontological Engineering* (AAAI-97), Spring Symposium Series, pp. 138-148.

TANIAR, D. & RAHAYU, J.W. 2006. *Web semantics & ontology*. London: Idea Group Publishing.

TARNAS, R. 1991. *The passion of the western mind: Understanding the ideas that have shaped our world view*. London: Pimlico.

TRIM, P.R.J. & LEE, Y.I. 2004. A reflection on theory building and the development of management knowledge. *Journal of Management Decision*, vol. 42, no. 3-4, pp. 473-480.

TROCHIM, W.M. n.d. *The research methods knowledge base*, 2nd edition. [Online]. Available: http://www.socialresearchmethods.net/kb/ [Cited 7 April 2008].

TRUEX, D., BASKERVILLE, R. & TRAVIS, J. 2000. A methodological systems development: The deferred meaning of systems development methods. *Journal of Accounting Management and Information Technology*, vol. 10, no. 1, pp. 53-79.

TUCKER, W. 2009. It is not just a theory, it is a theory! *CAP Journal*, no. 5, pp. 9-10.

TURBAN, E., MCLEAN, E. & WETHERBE, J. 2004. *Information Technology for Management: Transforming organization in the digital economy, 4th ed*. New York, NY: John Wiley & Sons, Inc.

UNIVERSITY OF VICTORIA, n.d. *Criteria for assessing PhD thesis*. Faculty of Graduate Studies. [Online]. Available: http://web.uvic.ca/gradstudies/pdf/phdcriteria.pdf [Cited12 May 2010].

USCHOLD, M. & GRUNINGER, M. 2004. Ontology and semantics for seamless connectivity. *SIGMOD Record*, vol. 33, no. 4, December.

VAN HEIJST, G., SCHEREIBER, A.T. & WIELINGA, B.J. 1997. Using explicit ontologies in KBS development. *International Journal of Human and Computer Studies*, vol. 46, no. 2/3, pp. 293-310.

VAN NIEKERK, J.C. & ROODE, J.D. 2009. Glaserian and Straussian Grounded Theory: Similar or completely different? In *Proceedings of the 2009 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists, Vaal River, South Africa, October 13-14*, pp. 96-103.

W3C. 2004. *Resource Description Framework (RDF): Concepts and abstract syntax.* [Online]. Available: http://www.w3.org/TR/rdf-concepts/ [Cited 23 January 2009].

WALSHAM, G. 1993. *Interpreting information systems in organizations.* London, UK: John Wiley & Sons.

WAND, Y. & WEBER, R. 1990. Toward a theory of the deep structure of information systems. *Proceedings of the International Conference of Information Systems, Copenhagen*, pp. 61-71.

WAND, Y. & WEBER, R. 1993. On the ontological expressiveness of Information Systems analysis and design grammars. *Information Systems Journal*, vol. 3, no. 4, pp. 217-237.

WAND, Y. & WEBER, R. 2002. Research commentary: Information systems and conceptual modelling - a research agenda. *Journal of Information Systems Research*, vol. 13, no. 4, pp. 363-376.

WEBER, R. 2003. Conceptual modelling and ontology: Possibilities and pitfalls. Research paper review. *Journal of Database Management*, vol. 14, no. 3, pp. 1-20.

WHITTEN, J.L., BENTLEY, L.D. & DITTMAN, K.C. 2004. *Systems analysis and design methods,* 6th ed. Boston, MA: Irwin/McGraw-Hill.

WIKIPEDIA, n.d. *Ethnography.* [Online]. Available: http://en.wikipedia.org/wiki/Ethnography [Cited 20 August 2006].

WINTER, S. & TOMKO, M. 2006. Translating the web semantics of georeferences. In *Web Semantics and Ontology, edited by D. Taniar & Rahayu, J.* Hershey, PA: Idea Group Publishing, pp. 297-333.

WYSSUSEK, B. 2004. Ontology and ontologies in information systems analysis and design: A critique. In *Proceedings of the Tenth Americas Conference on Information Systems, New York, NY, August 5-8,* pp. 4303-4308.

YU, E. 1995. *Modelling strategic relationships for process engineering.* Toronto: University of Toronto (PhD thesis, Department of Computer Science). (Also Tech. Report DKBS-TR-94-6.)

YU, E.K.S. 1997. Towards modeling and reasoning support for early-phase requirements engineering. In *Proceedings of the 3rd IEEE Int. Symp. on Requirements Engineering (RE'97), Washington, DC, Jan. 6-8.*

ZÚÑIGA, G.L. 2001. Ontology: Its transformation from philosophy to Information Systems. *FOIS'01,* 17-19 October 2001, Ogunquint, Maine, USA.

APPENDICES

Appendix A: Data Gathering Interview Questions

An Ontological Approach to Software Development: PhD Interview Research Questions.

Brief Introduction:

All information systems that are developed consist of three basic components, that is, database, user interface, and applications. Linking all these three is a common schema as shown in Figure A1 below (adapted from Sowa, n.d.).

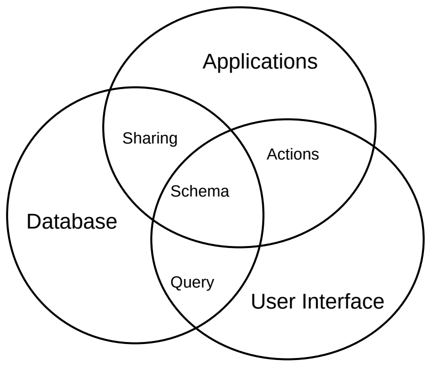


Figure A1: An Integrated Information System (*Adapted from Sowa, n.d.*)

The purpose of this research is to position ontology as an artefact that integrates all three, is part of each of the components and, at the same time, is a tool that can be used to develop evolvable software products with emergent properties. Such properties are the ones needed in fast-growing, interactive and evolving systems.

The research approach is a qualitative empirical investigation into the domain of software development. The researcher investigates practices around the development and use of software in information systems. An analysis of the technical design of the software and information systems is also part of the investigation.

Statement of the problem:

The problem that software developers continue to face is the lack of a method or tool that can augment current programming language technology and methodologies with semantics-based tools to enable construction of evolvable software products. Software kernels (Dittrich & Sestoft, 2005) and software product lines (Carnegie Mellon Software Engineering Institute, n.d.) have been used in industry, research, and academia to try to address the speed of reproducing and of the evolution of software products. Although they have gained widespread acceptance, they still leave a semantic gap that needs to be filled and incorporated in software products. An ontological approach to software development challenges most of the traditional software development approaches and borrows most of its characteristics from the agile methodologies family.

Aim of the Study

The aim of the study is to position ontology as an artefact that can be incorporated in software applications, either at run time or as a part of an information system, so as to increase semantic and context-awareness in information systems. This development will result in information systems that are more dynamic and usable with components that are reusable, shareable, visible and accessible to other systems and users as well.

Objectives of the Study

The objectives of the study can be summarized as the need to develop a general ontology framework that satisfies the aims of the study. The importance and validity of the framework will be justified using data gathered from the developers of information systems. More specifically the objectives of the study are:

a. To develop or construct a tangible ontological theory within the field of software development that can be used to design and develop romantic information systems. The abstract ontology concept will be operationalized into a concrete form of knowledge that can be applied readily in software development (Agerfalk, 2004). Operationalization is the process of converting or moving abstractions to concrete artefacts that have practical applications in a social practice, such as an inter-organizational system.

b. To develop a linguistic model that can be used by software agents to communicate in information systems.

c. To position ontology as an artefact that is needed by software agents to enable them to develop the linguistic models needed for communication in information systems, both experientially and intuitively.

d. To develop an ontology model-driven-approach to software development (El Baze, 2005). This approach will ensure that the resultant model moves from a purely abstract conceptualization of an ontology model to a truly operational model. This context-aware, purpose-specific actionable model could be deployed as a value-added service layer in software development processes.

Purpose:

The aim is to understand and explain the software development practices and designs from practitioners' view point. This aim is explained by – and based on – the historical and situational context of software development. The secondary aim of this research is to derive a new paradigm in systems development. The questions will be open and unstructured. In those situations that permit it, observation will be used to complement the data-gathering process.

Activities:

The researcher will investigate software development approaches, methods, techniques and tools currently used in software development. Considering the pervasiveness and ubiquity needed in current information systems, the activities will unravel the discrepancies between current development methods and the one proposed and strongly motivated by the literature study, that is: an ontology-driven development approach.

Section A: These few questions ask for demographic data

i. For identification purposes only, can you tell me your name or the name of the company you represent?

ii. How long have you been involved in software/system development?

iii. What type of software products or information systems does your company specialize in?

iv. What is your role in the company?

Section B: These questions look at the different software/system development practices from the participants' view point.

These questions generally look at the paradigm aspects of software development.

Question 1:

There are several software development paradigms in widespread use today.

a) Which paradigm do you advocate when developing software products in your organization?

b) What is the main reason why you use this paradigm?

Question 2:

Some people say the difference between the structured methodology (use of process and data models and use of DFD's and ERD's among other techniques) and the object oriented paradigm (use of use cases, classes and activity diagrams with UML as a technique) is that the latter captures semantics of a system. What is your opinion of that?

The researcher needs to find how context and meaning in systems is captured during development time and at run time of software products.

'There is a reason why computers have not yet become fervent natural language speakers. (It's not a matter of processing power and never will be): we simply are not programming them correctly.'

(El Baze, 2005.)

Question 3:

What is your comment on this?

Question 4a:

It is common knowledge that information systems do not behave the same way as humans do. What do you suggest should be done to add this human-like behaviour in information systems artefacts?

Question 4b:

Information systems comprise three parts, the formal, informal and the technical part. Can you explain how the formal and informal part is captured during your software development processes?

Question 5:

Organizations as systems exhibit a lot of "unstated" assumptions that are reflected through the shared (common sense) knowledge of people familiar with the social businesses and technical contexts within which the proposed system will operate (Rosenkranz and Holten, 2007:58) the pragmatics. What can be done in software development to capture these assumptions?

Question 6:

Every organization runs information systems within an organizational culture and context. When developing software products, how can developers ensure that this organizational culture and context are captured and implemented in the software products?

The next group of questions looks at communication problems during software development.

Question 7:

a) When communicating with clients, what communication tools and channels do you use in your organization?

b) "It is easier for clients to understand the functionality of the software through the user interface sketches" (Dobing and Parsons, 2007:123).What is your view?

Question 8:

During software development, how can we improve the clients' understanding of the functionality of the proposed software?

Question 9:

In which stages of software development would you involve the client?

Question 10:

How do you ensure that the understanding of the client of what the system should do is correct and maintained throughout the development process from analysis to maintenance stages?

Question 11:

Communication is a very important facet of software development process. Many people decry the lack of a communication language/tool that is understandable by both users (clients) and developers. What tools can be used to communicate between client-analyst and between analyst-programmer during software development?

Question 12: *(Check question 7 for similarity in answers)*

What type of clients do you have? Do they have knowledge of system development or software development tools?

Hint: What do you use to communicate with these clients?

The next group of questions looks at the ability of ontology-driven case tools to retain the system context gathered during the analysis to maintenance processes.

Question 13:

What do you think about a software development tool that integrates the software development tasks and data or information gathered from analysis through to implementation can improve on the software development process?

Question 14:

It is recommended that, when you develop a software development approach or methodology, one should also develop an accompanying development tool. The tool has considerable influence on how the approach or methodology will subsequently be used. What functions or tasks would you want to see in a development tool that supports a development approach/ methodology that allows adaptability and reusability in software products?

Question 15:

What is your understanding of adaptive software products?

Question 16:

How can you ensure the development of adaptive software products?

Question 17:

a. What is your understanding of evolving software products or systems?

b. How can you ensure the development of evolvable products?

This group of questions investigates, inter alia, issues affecting software development productivity. Ontology has the capability to improve software productivity, as will be discussed in the thesis.

Question 18:

How do you ensure requirements gathered during analysis and design specifications are reused?

Question 19:

As a software developer, you have developed software products for different organizations that are in the same industry. How do you reuse knowledge gained from one development process to the next project?

Techniques that mediate between a dialect conflict of context and meaning.

Question 20:

The success of any software development process depends on the ability to create a language community among stakeholders. This leads to the creation of shared knowledge that will later be captured in the resultant software product or system. The shared knowledge reflects itself through the concepts that will be used in a specific domain. How would you, as a developer, mediate when working in the same industry, if the same concept is given different meanings? That is, there exists a dialect conflict of context and meaning (context versus meaning).

Software success metrics looking at the quality of software products.

Question 21:

How do you ensure that the software development process is?

i. Within budget

ii. On time

iii. Easy to modify

iv. Faulty free

v. User is satisfied.

Question 22:

How do you manage scope creep (handle) at the same time ensuring all of the above?

The following questions look at design problems that can be handled using ontology. The technical nature of the ontology artefact requires questions to be very general and the research extrapolates the technical responses at the data analysis stage.

Question 23:

How do you capture semantics (meaning) of designs, specifications in your systems?

Question 24:

How do you ensure that systems are developed within a specific application (domain) context?

Question 25:

How can we improve the changing and communication of a software product design change during coding time?

Question 26:

Discuss some of the challenges that you have encountered pertaining to developing software products that are reusable.

Hints for Interviewer:

- *Design for reuse*
- *Cost of reusing the components*
- *Archiving and retrieval of reusable components*
- *Accessibility of the source code - if not home grown.*
- *Adaptability and evolvability of source code components*
- *Any other / maintainability*
- *Progress measure during software maintainability / development.*

<u>End of Interview</u>

Appendix B: Publications Generated from the Research Process

a. MAVETERA, N. 2011. Towards an Ontology-Driven Software development Approach: In *Proceedings of the 16th IBIMA Conference on e-Government,* Innovation and Knowledge Management: A Global Competitive Advantage, Kuala Lumpur, Malaysia, June 25-July 2, pp.1050-1057.

b. Mavetera, N. 2011. The use of GTM in Qualitative Research. *In Innovative Teaching, learning and Research Methods in Higher Education, Book Chapter, L. Makondo & MA. Mokoena Eds., And Cork Publishers, 2011. To Appear.*

c. Kroeze, J.H., Lotriet, H.H**.,** Mavetera, **N**., Pfaff, M.S., Postma, D.J.V.R., Sewchurran, K. & Topi, H. 2011. ECIS 2010 Panel Report: Humanities-Enriched Information Systems. *Communications of the Association for Information Systems Journal (CAIS)*, vol. 28, no. 1, article 24. Available at: http://aisel.aisnet.org/cais/vol28/iss1/24.

d. MAVETERA, N. & KROEZE, J.H. 2010a. Considerations in Grounded Theory Research Method: A Reflection on the Lessons Learned. In *Proceedings of the 14th International Business Information Management Association Conference (14th IBIMA), Business Transformation through Innovation and Knowledge Management: An Academic Perspective*, *Istanbul, Turkey, June 23-24*, edited by Khalid S. Soliman. pp. 1454 - 1469.

e. MAVETERA, N. & KROEZE, J.H. 2010b .An Ontology-Driven Software Development Framework. In *Proceedings of the 14th International Business Information Management Association Conference (14th IBIMA), Business Transformation through Innovation and Knowledge Management: An Academic Perspective*, Istanbul, Turkey, June 23-24, edited by Khalid S. Soliman. . pp. 1713 - 1724.

f. MAVETERA, N. & KROEZE, J.H. 2010c. Guiding Principles for Developing Adaptive Software Products. *Communications of the IBIMA*, vol. 2010, Article ID 340296.

g. KROEZE, J.H.; LOTRIET, H.; MAVETERA, N.; POSTHMA, D.; PFAFF, M.; SEWCHURRAN, K. & TOPI, H. 2010. Humanities-enriched Information Systems; panel discussion paper In *Proceedings of the 18th European Conference on Information Systems (ECIS), IT to Empower, University of Pretoria, June 6-10* 2010.

h. MAVETERA, N. 2010. Philosophical Groundings in Social Science Research. In *Fundamentals of Scientific Writing and Publishing: Research and Publication Series Volume 1*, Mafikeng: The Platinum Press.

i. SHAWA, W; MAVETERA, N. & HUISMAN, M. 2009. Building Language Communities in Organizational System Development Teams Using Ontologies. *Proceedings of the 13th IBIMA Conference on Knowledge Management and Innovation in Advancing Economies,* Marrakech, Morocco, November 9-10.

j. MAVETERA, N. & KROEZE, J.H. 2009a. Issues Affecting Software Development: A South African Software Practitioners' Viewpoint. *Communications of the IBIMA*, vol. 9, no. 2, [Online]. Available: http://www.IBIMA.org. [Cited 25 April 2009].

k. MAVETERA, N. & KROEZE, J.H. 2009b. Issues Affecting Software Development: A South African Software Practitioners' Viewpoint. *In Proceedings of the International Business Information Management Association Conference (IBIMA),* Cairo, Egypt, January 4-6.

l. MAVETERA, N. & KROEZE, J.H. 2009c. Practical considerations in Grounded Theory Method Research. *Sprouts: Working Papers on Information Systems*, vol. 9, no. 32. [Online]. Available: http://sprouts.aisnet.org/9-32.

m. MAVETERA, N. & KROEZE, J.H. 2009d. A Grounding Framework for Developing Adaptive Software Products. In *Proceedings of the 13th IBIMA Conference on Knowledge Management and Innovation in Advancing Economies, Marrakech, Morocco, November 9-10.*

n. Mavetera, N. 2009. The Non Linearity Nature of Choosing a Research Method, *In Proceedings of the Postgraduate Supervision Conference, Stellenbosch University, April 27-30.*

o. MAVETERA, N. & KROEZE, J. H. 2008. Practical issues in Grounded Theory Method Research. *In Proceedings of the M & D Graduate Symposium, preceding SAICSIT 2008, Wilderness, October 6*Shawa, W. C.; Mavetera, N. & Huisman, M. 2008. An Ontology for Information Systems Development Methodologies. In Proceedings of the M & D Graduate Symposium, preceding SAICSIT 2008, Wilderness, October 6.

p. MAVETERA, N. 2004a. The Use of Ontologies in Designing Agents for E-Markets: From a Mechanist to a Romantic Approach. In *Proceedings of the International Business Information Management Conference (IBIM '04), Amman, Jordan, July 4-6.*

q. MAVETERA, N. 2004b. The Philosophy of Ontologies: A new Information Systems Development Paradigm. In *Proceedings of the International Science and Technology Conference (ISTC'04), Vanderbijlpark, December* 1-2 MAVETERA, N. 2007. A Comprehensive Ontology-Driven Software Development Architecture: A Holistic Approach to Developing Romantic Software Products, In *Managing Worldwide Operations and Communications with Information Technology (Proceedings of 2007*

Information Resources Management Association, International Conference, Vancouver, British Columbia, Canada, May 19-23,

Appendix C: Incidents for Proposition PA

In these appendices, the italicized phrases indicate a word or phrase that supports the hypothesis under discussion.

P 2: The *realistic world view says that everything is agreeable*. You and I can decide this is a mouse and this is the function of the mouse. That is *the realistic world view and that is where the problem with the engineers comes in*. They follow a *realistic world view*. They think they agree on the purpose of this thing.

P 2: And if we follow a *relativistic world view, you might have an opinion on this device and I can have a different opinion on this device.* And if we then get down to a discussion, we might design a much better mouse than this - if we sit down and think "what is the real purpose".

P 3: We shouldn't forget paradigms where they focus on people, *people-orientated paradigms*. We say, in the end, all the work we do, the systems we develop, the products we develop, is there to benefit people. So you should really be in contact with your users and so on. So that seems more *it's a social aspect* - your systems development.

Appendix D: Incidents for Proposition PB

P 1: So I think the best way to actually represent that information will be to come up with some - *not a framework* - *but maybe language* of some sort *that has the ability to actually express those concepts and to relate to technical issues that way*. I am tending to think in terms of the *approaches to development*.

P 1: Traditional approach, where you have a *systems analyst, you have a programmer*, and then you have got *the business guy, the user*. I call that the traditional approach. In the approach that I've studied in detail and also used, you'll find that *the gap between the analyst and the programmer is actually closed by coming up with somebody who's called a developer,* who actually *elicits the requirements directly from the user*. And that *user becomes part of the development team* which means that those details - those metaphors and the language that is used by the team becomes familiar to that user.

P 2: I'm very much *against the traditional paradigm* because I believe it leads to a case where you have *a system that conforms to the requirements and not to the user's needs.*

P 1: In *agile methodologies* - you'd actually *have the client from the beginning up to the end.*

P 2: But the best way is to *have a user onboard*, such as all these *agile methodologies*,

P 2: Object-oriented programming to me works well in the technical applications. Perhaps it's not yet proven enough for me in today's software development, except perhaps for requirements analysis.

P 2: *Object-oriented technique does allow more for people to capture all their requirements, to capture non-technical things.*

P 1: Soft issues- I would say [that] if an individual wants to work in that area, they'll have to look into a lot of other areas that may not necessarily be in IT but others are IT. For example going into psychology, going into organizational behaviour. Those are areas that are very, very interesting and there are books that people have written that bring the softer issues of IT and the technical issues together.

P 1: Language Requirement-Because the *language is the only solid thing* or it's the only *output that you get from a human being that lets you know what is happening in their brain.* And if you want to deal with the informal part of the system, you are really looking at the softer issues, *the human issues of software development*. The *behaviour of human beings* and *their languages would be the centre that people have to deal with in order to understand.*

P 3: Soft Systems Methodology - The other paradigm would be - I would call holistic paradigm where they focus on the system - the systems thinking.

P 3: I would advocate a *contingency approach*, meaning I would first of all go and look at the *type of project with the type of system* that I would have to develop, look at the *characteristics of this system* and then I would decide. Say for example it's a life critical system; you have to develop something for people in the hospitals or so. Then I would go for the *structured approach,* mainly *because it's very, very focused*; you know exactly what to do next. I would *adjust the structured approach a bit to make it iterative*, so you can *go back to previous stages* if you think there was a mistake there and because of the documentation.

P 3: In my research I saw that *those two are the mostly used approaches in the industry*. You find that there's a *movement towards rapid application development* but the *structured and the object-orientated approach* are *the most used approaches*.

P 1: In Extreme Programming you have a practice that is called Pair Programming. In Pair Programming you put two people - two programmers - on one machine and that way, *that knowledge which cannot be written down but which is in those developers' minds, can actually be exchanged as they do the development.*

P 1: But it should take a much longer study where you're trying to marry all those different fields because, if you are going to do such a task, you need to *know how people behave* and that's a different field from IT. So *the behaviour of human beings in different environments* - how do they do that.

P 1: Now that they [computers] can solve a good number of those problems, we now expect them to help us in thinking as well, so that we have a thinking machine that can solve some of your thinking problems. In that area, that's where I would say we are lacking. *We don't have the skills at the moment to actually come up with systems like that.* The reason being that *the human brain itself and the behaviour of the human being, those two issues are still grey* even to those who are specialists in those areas.

Appendix E: Incidents for Proposition PC

P 1: So one approach that should be used to actually capture that kind of [tacit] knowledge would be to allow in the systems development process *a [discussion] forum or some place within the process where that knowledge can be exchanged within the individuals who are involved in the development team.*

P 1: So these *whiteboards* would be - *that's where people would just write those terms that they are used to, that they normally use.* And then the *flyers* would actually - *when people have a meeting they talk and so forth - then you stick those along the corridors in that organization.*

P 1: User Involvement - You don't want the systems analyst to meet some guys there, get some documents signed and then go and shift that information into the programme, but *you want these three levels [the analyst, the programmer and the user] to actually sit together.* The *user must be the centre* of all those things.

P 1: Quality Measure - So the *client [user] must be there throughout and that is basically for quality checks purposes, to ensure that the kind of requirements that the user has are actually met.*

P 2: *Involving your users in all the stages,* so that we don't spend a lot of time on things that's not important.

P 1: So with user stories, *you'll ask the user to give you a different story of what functionality they would like to be implemented.* It's not as detailed as a *use case but you give them some 6 x 6 inch cards and ask them to write a little functionality that they want implemented.* So you're breaking down the requirements into those small user stories.

P 3: What they [developers] do, *they get the requirements on user cards*. You know the *user stories*? Normally they're the small cards like [...] this size - and then *the users would write their requirements here of the system.* And what they do, in the end *they collect all the cards and they put a price tag on it*. The price in terms of the *time* it would take and the *physical resources* they would need and the *money*. And then they tell the users "what would you like us to do in the following three weeks".

P2: *If you show them these prototypes or these interface sketches*, like you call it, that *they would understand it much better. To them the system is the interface.*

P 1: *Proof-of-concept* is more practical, where you're coming up with *a demo of the system*, how the system is going to work. That's basically what really impressed them because they could *now start contributing* and saying "okay, at this point we want this" or "at this point, no, we don't deliver this kind of reports.

P 2: A *prototype* tells you sometimes that - it's empty behind that screen and *the proof-of-concept has some functionality* and you should have a bit of functionality.

P 2: To do a *very good verification after the requirements has been gathered*, can be done, once again, with the *proof-of-concept*.

P 1: When *you look at an object, it gives a better understanding of the system…* but *I wouldn't call it semantics*. I think *it's lighter than semantics*. It doesn't give the details that semantics would give. It, however, *leaves you at a general level* and that is an *advantage of object-oriented programming*. It gives a general class you can actually inherit and start reusing in the system. So I would say *there's a level of understanding that has been increased, as opposed to structured programming* but they still need to go deeper which would now become the semantic level.

P 1: [Language Limitations] For example, *when somebody expresses their idea, the* *limitations of that expression or that representation, are within the language that they are using.*

P 3: [Communication problem between the conceptual and physical design.]

I would say maybe some of the environments can handle that but sometimes it's difficult. It's up to a certain stage. Once you've generated your physical database, it's difficult. So I agree, it could help but I would first of all really test the system or - no, not the system - the environment to see that it's done, it can handle it.

P 3: [Language Requirement] *Plain common language;* not technical jargon. Sometimes we don't even - we are not familiar with our own language, the terms we use, the slang that we use. So we should really, *really use just plain common language at the level that the user can understand.*

P 3: I think sometimes - you know, nowadays people are very interested in business analysts or business analytics or whatever they call it, where you have business people talking to your users. But I would go further, I wouldn't just have a business person there, I would like a technical person with business knowledge to do this, to do the communication

P 3: [Communication Problem] And I - really I would argue that it's because the client can't define the requirements correctly because we don't understand what exactly this user wants.

P 1: So that is exactly the demise that we have as software developers, that if we try to go into the details of the softer side - the non-formal parts of the system - we find that we have limitations in terms of the language. Because *whatever language you are using, when I try to use now the computer language to express that, it might not represent what you are talking about.* So it is really the issues of language that should be dealt with and so forth.

P 1: But what usually happens is that when you get to an organization, you need *to understand their language, the way they talk about things, whether they are technical or non-technical* but for the purposes of your requirements. The way the information is represented will sometimes differ from other organizations even in the same field.

P 2: The other way is to *incorporate the clients right through the process and not only in* *the requirements phase* - but to talk to clients right through.

P 2: You can use UML quite well. But *one should not stick to the rigidness of such techniques* or *the prescriptiveness*; one should *be able to add your own things*, like to be flexible. *We don't have enough space on those diagrams to write down what its purpose originally was, and that is to capture non-technical information.*

P 2: That's why *ERDs are so difficult and not a very good tool to use, except in the physical design*. It's *not a requirements validation tool* for end-users.

P 2: [Star schemas] are the *basic design in data warehousing*. They replace the ERD. They replace the ERD but they are designed in such a way that I can teach you within three seconds how a star schema works but I won't even attempt to teach anybody an ERD.

P 1: [schemas] So now what I'm trying to say is that you need to come up with some schemas because that language is full of these schemas that you're used to, for example, you're trying to come up with a system for the library. In order for you to represent the requirements there, you come up with those schemas that will give you, for example, the - you can have schema called "loaning a book" or something like that.

P 3: The *user interface sketches* that you talk about could be used [to communicate system understanding with the users]. If you show *users* these prototypes or *these interface sketches*, they would understand it much better. To them the *system is the interface*.

Appendix F: Incidents for Proposition PD

P 1: *Iterative development*", where you develop your system in small iterations, and at the end of each iteration, you actually deliver a part of the system that is actually working.

P 2: Incremental development helps a lot *to do delivery in smaller pieces,* and then if you go wrong you don't have to backtrack the whole system. So *iteration and an incremental development* help a lot.

P 1: So an adaptive application would be an application that is highly maintainable, where you can get into the different modules and components of the system, and you do maintenance, *put new additions into that system without overburdening the system*, without getting what is called “spaghetti cord” where the cord is no longer maintainable.

P 1: [Adaptive Products] should be products that, when you get into the maintenance phase, *they can easily be changed.* Software maintenance, basically it’s not about worn-out parts that you are replacing, it’s mainly about *bringing in the new changes that are relevant to the system in its environment.*

P 2: [Adaptive software products] means that the *product can change according to business needs* and it *should not take too long to make those changes.* And business needs can be requirements or it can be financial constraints or time constraints. If they run out of time you should be able to give them something at least usable. But mainly it should *be able to adapt to changes in business and business strategy within reasonable boundaries.*

P 3: [Adaptive software products] would be systems that can *change quickly according to changing environments and requirements*

P 1: [Software evolution] So his (this???) concept is that if you go [into] electrical engineering - a field of electrical engineering called *“control engineering*” - you’ll find that, in order to stabilise a system, they put all systems *in black boxes,* especially the processes - industrial processes. They’ll put it in a black box and say “a system has an input, and output”. And then, in order to stabilise that system, you are saying the output must be taped as either a part of the output or the entire output. You’ll tape it and fit it back into the input, so there must be a mixer there - some function which you represent by a summation sign to say “we are summing the input and part of the feedback”. But in so doing you are regulating the output so that the output does not spiral exponentially. Right, and that concept of spiralling exponentially you’ll find it - let’s say in speakers - if you put a microphone next to the speaker, then you have negative feedback. So you are trying to control the feedback there.

P 1: [Software Evolution] So in software evolution now the Lehman concept actually says that the moment you deploy the system and the user starts using the system, they’ll tell you some things that they don’t like about the system and whatever - things that need to be changed about the system. So that should be considered as feedback from the system that you have delivered. So *take that feedback and feed it together with the input, to control the way your system should actually work.* So that’s the concept. It’s trying to use that control engineering approach to actually regulate the way we develop our systems.

P 1: Because *software development is not like other IT fields, like traditional engineering* and so forth. You find that you can’t come up with blueprints and put them there and say [that] people are going to develop according to this, and they follow that because basically things *are based on the human brain,* it’s more like an art.

but I would say the object orientation really could be more meaningful.

P 3: [Development Problem] And I - really I would argue that it's because the (client can't??) define the requirements correctly because we don't understand what exactly what this user wants. So if we put more effort into requirements gathering, I think you will see that problem - I wouldn't say it'll be solved - but much better. I think *we are so much in haste to get the product delivered, that we don't even bother to look at the requirements.*

P 3: And another thing I would also be careful though is, if companies buy into one of those software engineering environments - say for example Oracle Designer - that software engineering environment is essentially a companion to a methodology. For example, Oracle Designer, the companion methodology I would say is like information engineering. They call it something else but in essence it's information engineering. And if that methodology is not understood and in place in the company, and you use this software engineering environment, it's difficult. They find it difficult to do that. But if they know the methodology and they buy into this software engineering environment, then it becomes (inaudible - speaking softly). I call a software engineering environment, a methodology companion, those should go together. You can't have the one without the other one.

P 3: [Language Community] And there should be users at our side when we develop these requirements, and we should give them examples.

P 3: [Developer Understanding of Business Model] So, once again - so you walk into a company and you have to find out what these assumptions are. That's the way. Once again, a deeper of knowledge of business and communication. I think that's a big problem for consultants, if they go into the business and they just have to do consultancy (inaudible - noise) straight answer in my opinion would be communication and, hang on a bit, just get to know the business. And after you've done that for a period, you can start with your work.

P 3: [Problems with case tools] So it could help. In my experience through my research, I discovered that sometimes the analysis takes longer than what the people or the developers were used to and then they get discouraged. They should just keep with the process, although it takes longer.

P 3: [Scope creep] Is spent on 20 percent of the requirement. Are they willing to do that? Are they willing to spend 80 percent of the time? If they cut that 20 percent of the requirement, they could save 80 percent of the time you can go onto a next project. So just discuss it with them. Ask them what their philosophy is, I would say, because some people really want that product and they want it in the way they visualise it and they want to add stuff and so on. And if they are willing to pay for that, fine with me, then I will add it because in the end of the day, it's [them] that are paying for the product, not me.

But other companies are very, very, very [results] driven and they want you to produce quick results and then I would show them. Think about it, if you are in your class also, the amount of time you spend with students in your office. It could be one student that takes up all your time in the office. It's the same with this principle. There's a small amount of requirement and it takes 80 percent of your time.

Appendix H: Full list of interview audio recordings (on CD)

Appendix I: Full list of interview transcripts (on CD)

Appendix J: Full list of incidents and codes for all the interviews (on CD)

Appendix K: Network diagram (on CD)

Appendix L: Full List of codes and code families for all the interviews (on CD)

Appendix M: Full list of incidents and memos for all the interviews (on CD)

Appendix N: Definition of Concepts Used in the Thesis

IT artefact: Orlikowski and Iacono (2001:1) refer to this as "bundles of material and cultural properties packaged in some socially recognizable form such as hardware and /or software". On the other hand, Hevner *et al.*, (2004:77) regard it as "constructs (vocabulary and symbols), models (abstractions and representations), methods (algorithms and practices) and instantiations (implemented and prototype systems)". In this research, however, the IT artefact takes into consideration the culturally and socially recognizable organizational attributes, which at the same time, should be bundled into abstract constructs, which exhibit and inform people's practices.

Approach: Checkland (1999) refers to an approach as a way of going about tackling a problem. However this approach may not be very prescriptive on the method to be used when tackling such a problem. (See *Section 4.7 for more detail.)*

Methodology: This can be regarded as a formalized approach or as a combination of steps and deliverables that are needed for the development of a software product (Dennis *et al.,* 2001). This, however, comes at a lower level of the hierarchy than the approach discussed above. Benson and Standing (2005:203) somewhat philosophically define systems methodology as a "series of steps that are used in solving a problem to a general approach to problem solving".

In contrast to an approach, a methodology as described includes the philosophical underpinning found in an approach, in which the "phases, procedures, rules, techniques, tools, documentation management and training for developers" are also listed (Benson & Standing, 2005:203). In research, however, a methodology can be conceived as a "model, which entails theoretical principles as well as a framework that provides guidelines about how research is done […] in the context of a particular paradigm" (Sarantakos, 2004:32).

Model: Aβmann *et al.* (2006) describe a model as a representation, description and specification of a thing. The sole purpose of creating models is to portray reality faithfully. Although models are abstractions of such reality, any manipulations or queries made using the model should be reliable adaptations of the actual thing that they represent. A model can also be viewed as a pattern, plan, representation (especially in miniature) or as a description designed to show the main object or workings of an object, system or concept.

Information Systems: From an academic point of view, Roode (1993) defined an information system as '*an interdisciplinary field of scholarly inquiry, where information, information systems and the integration thereof with the organizations are studied in order to benefit the total system (technology, people, organizations and society)*'. In practice, Turban *et al.* (2004:18-19) distinguish between formal and informal information systems but the overarching characteristic of these systems is their ability to "*collect, process, store, analyze and disseminate information for a specific purpose".*

Mechanistic Systems: These are systems that are very efficient at capturing and structuring data to enable and facilitate its interpretation. As Roode (1993) states, processing data and providing 'information' using technology is only the first step in a series of activities that lead to the provision of information as understanding. Mechanistic systems are based on the concept of explicit programming, which leaves the resultant system unable to provide its users with semantically rich information. The mechanistic notion assumes that the world is rational, deterministic and can be described sufficiently using rules and algorithms.

As Struwig and Stead (2004) state, the mechanistic world view conceives reality as existing and a given. Mechanistic software products are, therefore, developed using this notion of the mechanistic world view.

Romantic Systems: These are systems that possess a certain degree of humanistic behaviour. They are open and non-deterministic and unlike machines, do not subscribe to mechanistic ideas of representation, formalization, program, order, reason, stability or control. Their behaviour incorporates the importance of power struggles, changing practices and impending chaos. They borrow their definition from romanticism (Basden, 2001; John, 2003; Gregor, n.d. and Loflin, n.d.), which may imitate belief systems that depend on 'irrationalism and feelings'.

According to Struwig and Stead (2004), the romantic world view posits reality as a social construction and believes in a shared reality among actors. Most importantly, what may be considered as knowledge depends heavily on the context, organizational politics and cultural factors as discussed in Chapter 4. Romantic software products should, therefore, be developed using this notion of romanticism. However this notion has practical limitations as will be discussed in this thesis.

Ontology: The ontology of a system can be described by defining a set of representational terms in the universe of discourse (Studer *et al.,* 1998; Swartout *et al.*, 1997; Mavetera, 2004a). It is an artefact that represents the acquisition, storage and dissemination of knowledge in information systems. In ontology, definitions associate the names of entities in the universe of discourse with human-readable text describing what the names mean and with formal axioms that constrain the interpretation and well-formed use of these terms.

Appendix G: Incidents for Proposition PE

P 2: [Requirements Repository] I don't think the end-user will benefit as much as the developer from doing it because you have one version of the truth of the requirement, you have one place - central repository - where you can find the requirements, you can set up checklists to see if you have fulfilled the requirements, the requirements are nearby but then the system must force the developers or the analysts to fill it in.

P 2: [Knowledge reuse] We call it experience. One thing one can say is that *involvement of the right people from the start.* If you've written a system and you - normally when it fails it is kind-of because that the right people was not involved from the company. That's my experience that the right people were not involved from the start and then that kind of experience I will take along and do a readiness assessment before I do anything else, and get the right people involved before I start working again on a new project.

P 2: [Scope creep] I think scope creep always happens but the extent of it depends on your - two things - actually mainly on the quality of your requirements. If your requirements are done well and *you involve the right people, you might have a situation where you don't get that much scope through it.* And then as - if you are using a *methodology where your end-user is involved, then scope should be their problem,* not your problem in a sense. But that's only the case if you have an end-user or a - I don't want to use the word end-user; I want to use a word sponsor user in data warehousing, *business sponsors* - somebody higher up in the organization. Somebody more important than the secretary that's going to type on the system; somebody that has power inside the organization. I want to have that person involved and he should prioritise all these additional scope requirements, not the programmers.

P 3: [Ambiguity on methodology that captures semantics] I think if you really, you do your ERs, your diagrams, and your dataflow diagrams, if you do that correctly and you explain it correctly to your user, I would say there could be semantics. Let me just think about it - that question. The object orientation, if you look at the use case diagrams, and you get the documentation of all the use cases, okay that could be more meaningful in a sense. There could be some semantics in there. Okay, I would say both

Framework: A framework is a conceptual structure that can be used to solve problems. An ontological framework, therefore, proposes ontology components that should be incorporated into software development practices.

Paradigm: According to Sarantakos (2004:31), a paradigm can be considered as a "set of propositions that explain how the world is perceived, it contains a world view, a way of breaking down the complexity of the real world". It is a holistic aggregation or representation of the values, beliefs and techniques that are shared by a specific scientific community. The same paradigm dictates the types of problems and solutions to these problems that can be investigated and how they are investigated.

Theory: In this study, it will be regarded as a soft, fluid set of agreed thinking, reached by similar-minded actors in an application domain. This agreed thinking makes up a coherent body of knowledge that can guide an actor to execute a certain action in a social practice (Agerfalk, 2004). This is also supported by Gregor (2006), Tucker (2009) and Sutton and Staw (1995).

Appendix J: Full list of incidents and codes for all the interviews (on CD)

HU: An Ontological Approach to Software Development
File: [C:\Documents and Settings\NMavetera\My Docum...\An Ontological Approach to Software Development.hpr5]
Edited by: Super
Date/Time: 10/09/03 05:40:09 PM

Code: adaptive evolutionary approach.. {1-0}

P 5: [adaptive evolutionary approach..] (121:121)   (Super)
Codes: [adaptive evolutionary approach..]

adaptive evolutionary approach,

Code: Adaptive software products. {5-0}

P 1: [It should be products that whe..] (115:115)   (Super)
Codes: [Adaptive software products.]

It should be products that when you get into the maintenance phase, they can easily be changed. Software maintenance, basically it's not about worn-out parts that you are replacing, it's mainly about bringing in the new changes that are relevant to the system in its environment.

P 1: [So an adaptive application wou..] (115:115)   (Super)
Codes: [Adaptive software products.]

So an adaptive application would be an application that is highly maintainable, where you can get into the different modules and components of the system, and you do maintenance, put new additions into that system without overburdening the system, without getting what is called " sphagheti cord" where the cord is no longer maintainable, it's growing beyond whatever and so forth.

P 2: [It means that the product can ..] (83:83)   (Super)
Codes: [Adaptive software products.]

It means that the product can change according to business needs and it should not take too long to make those changes. And business needs can be requirements or it can be financial constraints or time constraints. If they run out of time you should be able to give them something at least usable. But mainly it should be able to adapt to changes in business and business strategy within reasonable boundaries.

P 3: [It would be systems that they ..] (168:168)   (Super)
Codes: [Adaptive software products.]

It would be systems that they change quickly according to changing environments and requirements

P 8: [an application should be … lik..] (164:164)   (Super)
Codes: [Adaptive software products.]

Code: Adaptive Systems {2-0}

P 5: [heuristic systems that could l..] (79:79)   (Super)

Codes: [Adaptive Systems] [heuristic systems]

heuristic systems that could learn from the way that a person uses the system, and sort-of adapt its response to become more and more inline with how that person uses it.

P 8: [It means we should leave room ..] (160:160)   (Super)
Codes: [Adaptive Systems]

Code: Adaptive systems development approaches {3-0}

P 5: [In a crafted quality or agile ..] (119:119)   (Super)
Codes: [Adaptive systems development approaches] [timebox approach]

In a crafted quality or agile environment, I certainly don't make promises about things like that, and I say "give me a timebox, let's set some goals and objectives for that timebox and we will do the best we can". And at the end of the timebox we will review what we have achieved. So I can certainly ensure that the budget is adhered to because the timebox gives me a certain budget. I can achieve on time because the timebox is limited, say to six weeks, two months, three months, whatever; but I can't guarantee that I what I build will be easy to modify, fault free, and that the user will be satisfied. That we will see - on reflection see to what extent we have achieved that. If the customer says "you've achieved enough to gain my confidence" then we get another timebox to improve those things.

P 8: [So, like I said, it's basicall..] (112:112)   (Super)
Codes: [Adaptive systems development approaches] [Requirements of a new development approach] [Romantic Worldview]

P 8: [In comparison to an applicatio..] (162:162)   (Super)
Codes: [Adaptive systems development approaches]

Code: Advantages of Agile Approach {2-0}

P 7: [I think for now Agile System D..] (41:41)   (Super)
Codes: [Advantages of Agile Approach]

P 7: [I would say Agile System Devel..] (45:45)   (Super)
Codes: [Advantages of Agile Approach]

Code: Agile Approach Concept {2-0}

P 5: [crafted quality". Now crafted ..] (113:113)   (Super)
Codes: [Agile Approach Concept] [crafted quality"]

crafted quality". Now crafted quality is very much what most people call agile (inaudible - cross-talking) okay; and controlled quality is very much the kind-of conventional waterfall lifecycle approach.

P 9: [Agile method because everyone ..] (25:25)   (Super)
Codes: [Agile Approach Concept]

Code: agile approach. {1-0}

P 5: [But in situations where the so..] (1:1)   (Super)
Codes: [agile approach.]

But in situations where the solution can't be well-defined - maybe it's more of an experiential or research type of

initiative or where there's a tremendous amount of pressure to deliver results rapidly and therefore there isn't enough time to do thorough planning - then I would recommend more of an agile approach
----

Code: agile methodologies {4-2}

P 1: [agile methodologies] (25:25)   (Super)
Codes: [agile methodologies]

agile methodologies

P 1: [in the agile methodologies - y..] (107:107)   (Super)
Codes: [agile methodologies]

in the agile methodologies - you'd actually have the client from the beginning up to the end.

P 2: [But the best way is to have a ..] (63:63)   (Super)
Codes: [agile methodologies]

But the best way is to have a user onboard, such as all these agile methodologies,

P 2: [extreme programming and new de..] (67:67)   (Super)
Codes: [agile methodologies]
Memos: [User understanding]

extreme programming and new development,

Code: Agile System Development Lifec.. {1-0}

P 7: 08072102 Edwin.rtf - 7:3 [Agile System Development Lifec..] (35:35)   (Super)
Codes: [Agile System Development Lifec..]

Code: Alpha testing {1-0}

P 9: 08081800 Kabelo.rtf - 9:19 [So with alpha, you'll use your..] (41:41)   (Super)
Codes: [Alpha testing]

Code: Ambiguity on methodology that captures semantics {1-1}~

P 3: 07111501 Magda.rtf - 3:10 [I think if you really, you do ..] (57:57)   (Super)
Codes: [Ambiguity on methodology that captures semantics]

I think if you really, you do your ERs, your diagrams, and your dataflow diagrams, if you do that correctly and you explain it correctly to your user, I would say there could be semantics. Let me just think about it - that question. The object orientation, if you look at the use case diagrams, and you get the documentation of all the use cases, okay that could be more meaningful in a sense. There could be some semantics in there. Okay, I would say both but I would say the object orientation really could be more meaningful.

Code: band width value analysis, {1-0}

P 9: 08081800 Kabelo.rtf - 9:8 [band width value analysis,] (9:9)   (Super)
Codes: [band width value analysis,]

Code: Beta testing {1-0}

P 9: 08081800 Kabelo.rtf - 9:20 [beta is involved - whoever is ..] (41:41)   (Super)
Codes: [Beta testing]

Code: brainstorming forum {1-1}

P 1: 07110506 Ernest.rtf - 1:10 [brainstorming forum] (27:27)   (Super)
Codes: [brainstorming forum]

brainstorming forum

Code: bundle value analysis {1-0}

P 9: 08081800 Kabelo.rtf - 9:55 [bundle value analysis] (173:173)   (Super)
Codes: [bundle value analysis]

Code: Business analyst as a project manager {1-0}

P 9: 08081800 Kabelo.rtf - 9:43 [in terms of Ned Bank, the busi..] (103:103)   (Super)
Codes: [Business analyst as a project manager]

Code: business analysts {3-1}~

P 3: 07111501 Magda.rtf - 3:18 [business analysts] (84:84)   (Super)
Codes: [business analysts]

business analysts

P 7: 08072102 Edwin.rtf - 7:9 [The business analyst needs to ..] (65:65)   (Super)
Codes: [business analysts]

P 9: 08081800 Kabelo.rtf - 9:4 [business analyst] (9:9)   (Super)
Codes: [business analysts]

Code: business requirement specifica.. {1-0}~

P 9: 08081800 Kabelo.rtf - 9:32 [Meaning before we can, even st..] (79:79)   (Super)

Codes: [business requirement specifica..]

Code: Business Requirements Document {1-0}

P 9: 08081800 Kabelo.rtf - 9:3 [Whereby we'll take the busines..] (7:7)   (Super)
Codes: [Business Requirements Document]

Code: business sponsors {1-1}

P 2: 07111500 Goede.rtf - 2:29 [business sponsors] (103:103)   (Super)
Codes: [business sponsors]

business sponsors

Code: Capturing Human Behavior {2-2}

P 1: 07110506 Ernest.rtf - 1:53 [But it should take a much long..] (143:143) (Super)
Codes: [Capturing Human Behavior] [Humanist Requirement for Software Development]

But it should take a much longer study where you're trying to marry all those different fields because if you are going to do such a task, you need to know how people behave and that's a different field from IT. So the behaviour of human beings in different environments - how do they do that.

P 1: 07110506 Ernest.rtf - 1:54 [Now that they can solve a good..] (147:147) (Super)
Codes: [Capturing Human Behavior]

Now that they can solve a good number of those problems, we now expect them to help us in thinking as well, so that we have a thinking machine that can solve some of your thinking problems. In that area, that's where I would say we are lacking. We don't have the skills at the moment to actually come up with systems like that. The reason being that the human brain itself and the behaviour of the human being, those two issues are still grey even to those who are specialists in those areas.

Code: Capturing Pragmatic Knowledge {3-1}~

P 1: 07110506 Ernest.rtf - 1:1 [there are different approaches..] (23:23) (Super)
Codes: [Capturing Pragmatic Knowledge]

there are different approaches that people have tried to come up with that will assist in terms of capturing that kind of pragmatic knowledge, if you want to call it. Actually some people call it tacit knowledge. So one approach that should be used to actually capture that kind of knowledge, would be to allow in the systems development process a forum or some place within the process where that knowledge can be exchanged within the individuals who are involved in the development team.

P 1: 07110506 Ernest.rtf - 1:7 [In Extreme programming you hav..] (25:25) (Super)
Codes: [Capturing Pragmatic Knowledge] [Knowledge Sharing]

In Extreme programming you have a practice that is called Pair Programming. In Pair Programming you put two people - two programmers - on one machine and that way, that knowledge which can not be written down but which is in those developers' minds, can actually be exchanged as they do the development.

P 1: 07110506 Ernest.rtf - 1:9 [So in these informal meetings ..] (27:27) (Super)
Codes: [Capturing Pragmatic Knowledge] [Knowledge Sharing]

So in these informal meetings people discuss the technology, discuss the process and the type of work that they're doing. And you task especially the upcoming youngsters within the organisation to actually take down the information that people are bringing. And you also ask them to access some journals and so forth, and try to relate that information to the standard practices that people are following.

Code: Capturing Semantics Issue {3-0}

P 1: 07110506 Ernest.rtf - 1:57 [So that is exactly the demise ..] (155:155) (Super)
Codes: [Capturing Semantics Issue] [Language Limitations]
Memos: [ME - 11/13/08 [5]]

So that is exactly the demise that we have as software developers, that if we try to go into the details of the softer side - the non-formal parts of the system - we find that we have limitations in terms of the language. Because whatever language you are using, when I try to use now the computer language to express that, it might not represent what you are talking about. So it is really the issues of language that should be dealt with and so forth.

P 3: 07111501 Magda.rtf - 3:37 [Object-orientation, I think, i..] (232:232) (Super)
Codes: [Capturing Semantics Issue] [object-oriented approach,]

Object-orientation, I think, if you look at the instantiation of a class, then I would say that could - object would be the - ja, the semantics behind the design. You design the class and then an instantiation of the class, the object would be the semantics.

P 9: 08081800 Kabelo.rtf - 9:31 [you get that from conceptual, ..] (69:69) (Super)
Codes: [Capturing Semantics Issue]

Code: Case Tool {1-0}

P 5: 07121401-Barry Myburgh Corrected.rtf - 5:19 [If the customer that you are b..] (71:71) (Super)
Codes: [Case Tool] [software engineering environme..]

If the customer that you are busy working for is the kind of customer who understands the importance of giving you enough time to do the job properly, then the use of case tools can greatly improve the software development process.

Code: Challenge for Software development {1-0}

P 5: 07121401-Barry Myburgh Corrected.rtf - 5:9 [And therefore I suppose I do a..] (27:27) (Super)
Codes: [Challenge for Software development] [Lack of humanist
understanding]

And therefore I suppose I do agree that they're not programming them correctly but I'm not sure that it is possible to get that point where the computer can be as capable of understanding human speech the way that human beings can understand it. I don't necessarily think that it's an achievable goal.

Code: change control {1-0}

P 5: 07121401-Barry Myburgh Corrected.rtf - 5:37 [In a controlled quality enviro..] (125:125) (Super)
Codes: [change control]

In a controlled quality environment I apply change control. I go through all the mechanisms of managing change requests, evaluating the impact of the change, getting the customer to either agree to postpone the change, or pay more money for it, or to take more time for it or whatever. So change control is the answer.

Code: Collaborative Approach {1-0}

P 5: 07121401-Barry Myburgh Corrected.rtf - 5:17 [Obviously the most effective w..] (57:57) (Super)
Codes: [Collaborative Approach] [User Involvement]

Obviously the most effective way of trying to ensure that is if we have the client participate in the activities of the project. In other words, a collaborative approach where the client is really a part of the team.

Code: Communication Method {1-4}

P 8: 08072200 and 08072201-Sindiso.rtf - 8:18 [There's a system called Test D..] (74:74) (Super)

Codes: [Communication Method] [Test Director]

Code: Communication Problem {5-2}~

P 1: 07110506 Ernest.rtf - 1:31 [INT: I've met - in fact I've m..] (73:73)   (Super)
Codes: [Communication Problem]

INT: I've met - in fact I've met two types of clients. One group of clients were clients who really don't know much about systems development. For example, one system we're actually developing, a school management system, and you're talking about people like school administrators and teachers and so forth; so most teachers would not know anything about systems developments. They will just tell you about their curriculum and the kind of day-to-day administrative tasks and chores that they actually have; and maybe try and interpret that into a system.

P 3: 07111501 Magda.rtf - 3:12 [And I - really I would argue t..] (66:66)   (Super)
Codes: [Communication Problem]

And I - really I would argue that it's because the (client can't??) define the requirements correctly because we don't understand what exactly what this user wants.

P 9: 08081800 Kabelo.rtf - 9:44 [But now you find out that he m..] (105:105)   (Super)
Codes: [Communication Problem] [Problems of not involving the users]

P 9: 08081800 Kabelo.rtf - 9:49 [But now in terms of communicat..] (119:119)   (Super)
Codes: [Communication Problem] [Development Problem]

P 9: 08081800 Kabelo.rtf - 9:50 [So meaning communication, the ..] (123:123)   (Super)
Codes: [Communication Problem]

Code: Communication problem between the conceptual and physical design. {1-1}~

P 3: 07111501 Magda.rtf - 3:36 [I would say maybe some of the ..] (220:220)   (Super)
Codes: [Communication problem between the conceptual and physical design.]

I would say maybe some of the environments can handle that but (chuckling) sometimes it's difficult. It's up to a certain stage. Once you've generated your physical database, it's difficult. So I agree, it could help but I would first of all really test the system or - no, not the system - the environment to see that it's done, it can handle it.

Code: Communication Technique {4-18}

P 8: 08072200 and 08072201-Sindiso.rtf - 8:21 [The first one of course is jus..] (80:80)   (Super)
Codes: [Communication Technique]

08072201-Sindiso.rtf

P 8: 08072200 and 08072201-Sindiso.rtf - 8:23 [We also have meeting sessions ..] (80:80)   (Super)
Codes: [Communication Technique]

P 9: 08081800 Kabelo.rtf - 9:35 [Okay we use your normal UML to..] (87:87)   (Super)
Codes: [Communication Technique]

P 9: 08081800 Kabelo.rtf - 9:38 [In Nedbank they were using Mic..] (93:93)   (Super)

Codes: [Communication Technique] [Microsoft SharePoint]

Code: Communication Tool {1-2}

P 8: 08072200 and 08072201-Sindiso.rtf - 8:20 [emails] (80:80)   (Super)
Codes: [Communication Tool]

Code: component development {1-1}~

P 3: 07111501 Magda.rtf - 3:29 [component development] (172:172)   (Super)
Codes: [component development]

component development

Code: Concept negotiation Technique {3-1}~

P 1: 07110506 Ernest.rtf - 1:14 [So instead of people being stu..] (33:33)   (Super)
Codes: [Concept negotiation Technique] [Language Community]
Memos: [ME - 11/12/08 [1]]

So instead of people being stuck to those technical terms and so forth, they come up with some naming convention that is common to everyone. And that naming convention brings everyone to the same language, despite the differences and so forth.

P 5: 07121401-Barry Myburgh Corrected.rtf - 5:27 [the reality centre and modelli..] (101:101)   (Super)
Codes: [Concept negotiation Technique] [Language Requirement]

the reality centre and modelling things with everybody busily participating in the modelling that takes place, that to me is a very effective way of doing that mediation. Because if you are busy developing something like a mind map, you know, you type in the words that you think describe the point that has been made and the people who had made the point immediately see the words you use.

P 8: 08072200 and 08072201-Sindiso.rtf - 8:48 [Okay, how we do it at the mome..] (144:144)   (Super)
Codes: [Concept negotiation Technique]

Code: configurable systems {1-0}~

P 5: 07121401-Barry Myburgh Corrected.rtf - 5:23 [configurable systems] (87:87)   (Super)
Codes: [configurable systems]

configurable systems

Code: contingency approach, {1-1}~

P 3: 07111501 Magda.rtf - 3:7 [I would advocate a contingency..] (37:37)   (Super)
Codes: [contingency approach,]

I would advocate a contingency approach, meaning I would first of all go and look at the type of project with the type of system that I would have to develop. Look at the characteristics of this system and then I would decide. Say for example it's a life critical system; you have to develop something for people in the hospitals or so. Then I would go for the structured approach, mainly because it's very, very focused; you know exactly what to do next. I would adjust the structured approach a bit to make it iterative, so you can go back to previous stages if you think there was a mistake there and because of the documentation. So then I would advocate that.

Code: Controlled quality approach requirements {1-0}

P 5: 07121401-Barry Myburgh Corrected.rtf - 5:33 [So if I'm in a situation where..] (115:115) (Super)
Codes: [Controlled quality approach requirements]

So if I'm in a situation where I have to - facing penalties of something - if I have to deliver within budget on time, I will demand to follow a controlled quality approach. And then that means I must be given sufficient time to have done sufficient analysis prior to the contract so that I really understand what it is I have to do, and can I plan accordingly, and I can form a team of people who can deliver on those requirements.

Code: controlled quality" {1-0}

P 5: 07121401-Barry Myburgh Corrected.rtf - 5:31 [The one I call "controlled qua..] (113:113) (Super)
Codes: [controlled quality"] [waterfall approach]

The one I call "controlled quality"; the other one I call "crafted quality". Now crafted quality is very much what most people call agile (inaudible - cross-talking) okay; and controlled quality is very much the kind-of conventional waterfall lifecycle approach.

Code: crafted quality" {1-0}

P 5: 07121401-Barry Myburgh Corrected.rtf - 5:32 [crafted quality". Now crafted ..] (113:113) (Super)
Codes: [Agile Approach Concept] [crafted quality"]

crafted quality". Now crafted quality is very much what most people call agile (inaudible - cross-talking) okay; and controlled quality is very much the kind-of conventional waterfall lifecycle approach.

Code: data-orientated paradigm {1-2}

P 3: 07111501 Magda.rtf - 3:2 [And the data-orientated paradi..] (25:25) (Super)
Codes: [data-orientated paradigm]

And the data-orientated paradigm is where they say [that] data stays constant, so we should rather focus on the data instead of the process, it would make our lives much easier.

Code: decision-logic-table {1-0}

P 9: 08081800 Kabelo.rtf - 9:11 [decision-logic-table] (9:9) (Super)
Codes: [decision-logic-table]

Code: Developer Understanding of Business Model {2-1}~

P 1: 07110506 Ernest.rtf - 1:38 [And besides that, when you do ..] (103:103) (Super)
Codes: [Developer Understanding of Business Model] [Knowledge Sharing]
Memos: [Studying the business environment]

And besides that, when you do such workshops, you are not just aiming at them understanding the system but you're also aiming at the developer's understanding of the business model that the organisation is using.

P 3: 07111501 Magda.rtf - 3:20 [So, once again - so you walk i..] (108:108)   (Super)

Codes: [Developer Understanding of Business Model]

So, once again - so you walk into a company and you have to find out what these assumptions are. That's the way. Once again, a deeper of knowledge of business and communication. I think that's a big problem for consultants, if they go into the business and they just have to do consultancy (inaudible - noise) straight answer in my opinion would be communication and hang on a bit, just get to know the business. And after you've done that for a period, you can start with your work. But I don't think there's a luxury for that.

Code: Development Approach {1-8}

P 8: 08072200 and 08072201-Sindiso.rtf - 8:13 [I would prefer, … a method in ..] (68:68)   (Super)
Codes: [Development Approach] [New Development Approach]

Code: Development Method {0-13}

Code: Development Problem {9-5}~

P 1: 07110506 Ernest.rtf - 1:21 [And then you find that we don'..] (55:55)   (Super)
Codes: [Development Problem]

And then you find that we don't develop systems that actually directly mix the requirements of that particular individual because there's a gap between the understanding of how the system works from the core of the system - the methods and the procedures and so forth - and the actual interface that the user is actually using.

P 1: 07110506 Ernest.rtf - 1:22 [So you have the business that ..] (57:57)   (Super)
Codes: [Development Problem] [Translation Medium Requirement]

So you have the business that side, you have got the interface which the user is using, and you have the technical details of the system itself. So those are three different levels and you find that at those three different levels the user understands the business side and the interface, but does not really understand the technical details but the developer who is the one who has the technical details. So you find that that way now, those requirements will always be presented in the way that the technical guy understands which puts the user at a disadvantage.

P 3: 07111501 Magda.rtf - 3:13 [And I - really I would argue t..] (66:66)   (Super)
Codes: [Development Problem]
Memos: [Requirements problem]

And I - really I would argue that it's because the (client can't??) define the requirements correctly because we don't understand what exactly what this user wants. So if we put more effort into requirements gathering, I think you will see that problem - I wouldn't say it'll be solved - but much better. I think we are so much in haste to get the product delivered, that we don't even bother to look at the requirements.

P 3: 07111501 Magda.rtf - 3:16 [I don't know about you Nehemia..] (72:72)   (Super)
Codes: [Development Problem]
Memos: [Lack of trust]

I don't know about you Nehemiah but - there's sort-of a hostility between users and system developers. The users, they don't trust the developers and the developers talk down on the user. Okay, and that's something that's still alive and well and that bothers me. We are partners ?

P 8: 08072200 and 08072201-Sindiso.rtf - 8:26 [Because right now we can't do ..] (88:88)   (Super)

Codes: [Development Problem]

P 8: 08072200 and 08072201-Sindiso.rtf - 8:39 [there are many times where the..] (122:122)   (Super)
Codes: [Development Problem]

P 8: 08072200 and 08072201-Sindiso.rtf - 8:41 [Many a time analysts do not re..] (124:124)   (Super)
Codes: [Development Problem]

P 9: 08081800 Kabelo.rtf - 9:49 [But now in terms of communicat..] (119:119)   (Super)
Codes: [Communication Problem] [Development Problem]

P 9: 08081800 Kabelo.rtf - 9:51 [Because with the current - set..] (125:125)   (Super)
Codes: [Development Problem]

Code: Development Technique {0-10}

Code: Development Tool {0-3}

Code: differential testing, {1-0}

P 8: 08072200 and 08072201-Sindiso.rtf - 8:55 [differential testing,] (150:150)   (Super)
Codes: [differential testing,]

Code: Discussion Forum {2-1}~

P 1: 07110506 Ernest.rtf - 1:3 [So one approach that should be..] (23:23)   (Super)
Codes: [Discussion Forum]

So one approach that should be used to actually capture that kind of knowledge, would be to allow in the systems development process a forum or some place within the process where that knowledge can be exchanged within the individuals who are involved in the development team.

P 1: 07110506 Ernest.rtf - 1:12 [So these whiteboards would be ..] (31:31)   (Super)
Codes: [Discussion Forum] [Knowledge Sharing]

So these whiteboards would be - that's where people would just write those terms that they are used to, that they normally use. And then the flyers would actually - when people have a meeting they talk and so forth - then you stick those along the corridors in that organisation. So that as you walk up and down, you actually see those common terms and so forth
----

Code: document of inputs {1-0}~

P 8: 08072200 and 08072201-Sindiso.rtf - 8:47 [document of inputs] (144:144)   (Super)
Codes: [document of inputs]

Code: Documentation as a communication tool/method {1-0}~

P 5: 07121401-Barry Myburgh Corrected.rtf - 5:16 [But of course that is appropri..] (53:53)   (Super)
Codes: [Documentation as a communication tool/method]

But of course that is appropriate during the formatory (sp) stages of the project when you are busy driving out the understanding and so on. If you're getting more to the formal stages where you are saying [that] here is an artefact or a deliverable that is being packaged, and now needs to be reviewed and accepted, that's a totally different matter.

There I stick to the classical methods of distributing the documentation; everybody walks through the documentation and reads and approves what they've seen.
Code: Dooijeweert aspects {1-1}

P 2: 07111500 Goede.rtf - 2:33 [Dooijeweert aspects] (111:111)   (Super)
Codes: [Dooijeweert aspects]

Dooijeweert aspects

Code: Emphasis of Formal part {2-0}

P 5: 07121401-Barry Myburgh Corrected.rtf - 5:11 [you know, in our software deve..] (39:39)   (Super)
Codes: [Emphasis of Formal part]

you know, in our software development processes we tend to focus almost exclusively on the formal part. I've certainly not been involved in any information system development project where we've tried to create a computer that can sort-of adapt it's response to different needs without having to be changed. So - I mean, is that what you're sort-of suggesting? It's almost an artificial intelligence kind-of -

P 5: 07121401-Barry Myburgh Corrected.rtf - 5:12 [The systems that I'm involved ..] (43:43)   (Super)
Codes: [Emphasis of Formal part]

The systems that I'm involved with - have been involved with, have always been ones where we've said - we've specified the formal parts on how this thing must work and we build it to work that way, and that's what the customer wants.

Code: equivalence {2-0}

P 9: 08081800 Kabelo.rtf - 9:9 [equivalence] (9:9)   (Super)
Codes: [equivalence]

P 9: 08081800 Kabelo.rtf - 9:56 [use the classic equivalency.] (173:173)   (Super)
Codes: [equivalence]

Code: ERDs {1-2}

P 2: 07111500 Goede.rtf - 2:14 [That's why ERDs are so difficu..] (55:55)   (Super)
Codes: [ERDs]

That's why ERDs are so difficult and not a very good tool to use, except in the physical design. It's not a requirements validation tool for end-users.

Code: Evolvable Software Products Concept {4-1}

P 1: 07110506 Ernest.rtf - 1:50 [So basically evolving systems ..] (135:135)   (Super)
Codes: [Evolvable Software Products Concept]

So basically evolving systems are systems that don't remain doing what they were originally meant to do but these are systems where, based on the outputs that you get and the comments that the clients give you, you actually regulate - you maintain the system - you implement some changes in the system and allow that system to be adapted to the new environment that you have.

P 2: 07111500 Goede.rtf - 2:26 [this incremental development i..] (87:87)   (Super)
Codes: [Evolvable Software Products Concept]

this incremental development idea that we create a software package or product that is delivered in increments; and that you don't gather all the requirements (indiscernible - audio quality) from the starting step (indiscernible - audio quality) as you deliver, get more requirements and feedback, so that you have a better chance of designing a product that suits the real needs of the (indiscernible - audio quality).

P 3: 07111501 Magda.rtf - 3:30 [It grows over time. Evolve, ma..] (188:188)   (Super)
Codes: [Evolvable Software Products Concept]

It grows over time. Evolve, maybe I can draw a picture. I don't know if that's (inaudible - noise). What I would say is, you have your problem. Say for example that's your problem. And then I would normally - evolutionary development, I would divide it into certain chunks and then I would start. I would only implement that one - develop and implement. If it's spelled correctly, implement, I don't know. And then afterward you add something on. So that one would be in the organisation, they would use that. And then you build the second part, and you develop (indiscernible - audio quality) and implement. This is how I see evolutionary development, that with time it grows. So it's like incremental development, I would say. (indiscernible - audio quality) that's how I understand it.

P 5: 07121401-Barry Myburgh Corrected.rtf - 5:22 [Well that's very often the sit..] (83:83)   (Super)
Codes: [Evolvable Software Products Concept]

Well that's very often the situation where we don't have time to do the job thoroughly the first time we do it, so we try to deliver the first working version of the system as quickly as possible, and then we use that as a basis for evolving the system into the eventual scope and functionality that the customer really needs.

Code: Evolving Product development Technique {3-2}~

P 1: 07110506 Ernest.rtf - 1:27 [So you're breaking down the re..] (65:65)   (Super)
Codes: [Evolving Product development Technique]

So you're breaking down the requirements into those small user stories. Then you take those user stories now and ask the guy who is going to develop to now try and estimate how long it's going to take to actually develop that. Then you are coming up with some design already as you do that.

P 3: 07111501 Magda.rtf - 3:31 [incremental development] (188:188)   (Super)
Codes: [Evolving Product development Technique]

incremental development

P 3: 07111501 Magda.rtf - 3:32 [XP, the methodology called XP?..] (192:192)   (Super)
Codes: [Evolving Product development Technique] [Extreme programming]

XP, the methodology called XP? Xtreme Programming.

Code: Extreme programming {3-4}~

P 1: 07110506 Ernest.rtf - 1:5 [Extreme programming] (25:25)   (Super)
Codes: [Extreme programming]

Extreme programming

P 1: 07110506 Ernest.rtf - 1:15 [Extreme Programming -] (33:33)   (Super)

Codes: [Extreme programming]

Extreme Programming -

P 3: 07111501 Magda.rtf - 3:32 [XP, the methodology called XP?..] (192:192)   (Super)
Codes: [Evolving Product development Technique] [Extreme programming]

XP, the methodology called XP? Xtreme Programming.

Code: First step in developing software {1-0}

P 9: 08081800 Kabelo.rtf - 9:47 [I think if we can get the righ..] (117:117)   (Super)
Codes: [First step in developing software]

Code: functional analyst, {1-0}

P 9: 08081800 Kabelo.rtf - 9:5 [functional analyst,] (9:9)   (Super)
Codes: [functional analyst,]

Code: Functional Design Document {1-0}

P 9: 08081800 Kabelo.rtf - 9:33 [FDD] (51:51)   (Super)
Codes: [Functional Design Document]

Code: functional testing {2-0}

P 8: 08072200 and 08072201-Sindiso.rtf - 8:53 [functional testing] (148:148)   (Super)
Codes: [functional testing]

P 9: 08081800 Kabelo.rtf - 9:14 [We'll test the functionality b..] (11:11)   (Super)
Codes: [functional testing]

Code: heuristic systems {1-0}~

P 5: 07121401-Barry Myburgh Corrected.rtf - 5:21 [heuristic systems that could l..] (79:79)   (Super)
Codes: [Adaptive Systems] [heuristic systems]

heuristic systems that could learn from the way that a person uses the system, and sort-of adapt its response to become more and more inline with how that person uses it.

Code: Humanist Requirement for Software Development {3-0}

P 1: 07110506 Ernest.rtf - 1:51 [Because software development i..] (139:139)   (Super)
Codes: [Humanist Requirement for Software Development]

Because software development is not like other IT fields, like traditional engineering and so forth. You find that you can't come up with blueprints and put them there and say [that] people are going to develop according to this, and they follow that because basically things are based on the human brain, it's more like an art.

P 1: 07110506 Ernest.rtf - 1:53 [But it should take a much long..] (143:143)   (Super)
Codes: [Capturing Human Behavior] [Humanist Requirement for Software
Development]

But it should take a much longer study where you're trying to marry all those different fields because if you are going to do such a task, you need to know how people behave and that's a different field from IT. So the behaviour of human beings in different environments - how do they do that.

P 3: 07111501 Magda.rtf - 3:5 [we shouldn't forget is paradig..] (27:27)   (Super)
Codes: [Humanist Requirement for Software Development]

we shouldn't forget is paradigms where they focus on people, people-orientated paradigms. We say, in the end, all the work we do, the systems we develop, the products we develop, is there to benefit people. So you should really be in contact with your users and so on. So that seems more it's a social aspect - your systems development. And I think that's where your user interface; maybe it could link to that.

Code: impact analysis {1-0}

P 7: 08072102 Edwin.rtf - 7:1 [impact analysis sort-of on thi..] (25:25)   (Super)
Codes: [impact analysis]

Code: incremental approach {1-0}

P 9: 08081800 Kabelo.rtf - 9:16 [So meaning users will have to ..] (23:23)   (Super)
Codes: [incremental approach]

Code: Incremental development {2-2}

P 2: 07111500 Goede.rtf - 2:18 [Incremental development helps ..] (63:63)   (Super)
Codes: [Incremental development]

Incremental development helps a lot to do delivery in smaller pieces, then if you go wrong you don't have to backtrack the whole system. So an iteration and an incremental development helps a lot.

P 9: 08081800 Kabelo.rtf - 9:17 [So meaning users will have to ..] (23:23)   (Super)
Codes: [Incremental development]

Code: information engineering {1-1}~

P 3: 07111501 Magda.rtf - 3:26 [information engineering] (160:160)   (Super)
Codes: [information engineering]

information engineering

Code: inspections {1-0}

P 9: 08081800 Kabelo.rtf - 9:28 [inspections] (53:53)   (Super)
Codes: [inspections]

Code: integration testing {1-0}

P 8: 08072200 and 08072201-Sindiso.rtf - 8:51 [integration testing] (148:148)   (Super)
Codes: [integration testing]

Code: interactive development {1-1}

P 1: 07110506 Ernest.rtf - 1:41 [interactive development", wher..] (111:111)   (Super)
Codes: [interactive development]

interactive development", where you develop your system in small iterations, and at the end of each iteration, you actually deliver a part of the system that is actually working.

Code: intuitive system {1-0}

P 8: 08072200 and 08072201-Sindiso.rtf - 8:15 [So it's more of an intuitive s..] (70:70)   (Super)
Codes: [intuitive system]

Code: Issues to consider when choosing a methodology {1-0}

P 9: 08081800 Kabelo.rtf - 9:48 [I think if we can get the righ..] (117:117)   (Super)
Codes: [Issues to consider when choosing a methodology]

Code: Jad sessions {1-0}

P 9: 08081800 Kabelo.rtf - 9:25 [Jad sessions] (51:51)   (Super)
Codes: [Jad sessions]

Code: Knowledge Base {1-0}

P 5: 07121401-Barry Myburgh Corrected.rtf - 5:25 [I mean to give a - probably no..] (95:95)   (Super)
Codes: [Knowledge Base]

I mean to give a - probably not a very academic answer but in my personal life as at school - you know, pretty much as an independent consultant and contractor, I just keep everything that I've done and re-use whatever wherever I can. So I mean, it's actually not a matter of knowledge base that you can draw on and just re-use stuff and repackage it and stuff like that. So I mean that's really the short answer but it all depends on my memory of course, and my search engine's capability of finding the things at the right time. I mean, I literally on my notebook computer have got all the work I have done for the last 15 years is on that computer. So if I can remember that I've done it before for somebody else, and my desktop search engine can find it, then I'll re-use it. If I can't find it I don't re-use it. So it's really not a very good answer to your question.

Code: Knowledge reuse {1-0}~

P 2: 07111500 Goede.rtf - 2:27 [We call it experience. One thi..] (91:91)   (Super)
Codes: [Knowledge reuse]

We call it experience. One thing one can say is that involvement of the right people from the start. If you've written a system and you - normally when it fails it is kind-of because that the right people was not involved from the company. That's my experience, that the right people were not involved from the start and then that kind of experience I will take along and do a readiness assessment before I do anything else, and get the right people involved before I start working again on a new project.

Code: Knowledge Sharing {6-0}

P 1: 07110506 Ernest.rtf - 1:7 [In Extreme programming you hav..] (25:25)   (Super)
Codes: [Capturing Pragmatic Knowledge] [Knowledge Sharing]

In Extreme programming you have a practice that is called Pair Programming. In Pair Programming you put two people - two programmers - on one machine and that way, that knowledge which can not be written down but which is in those developers' minds, can actually be exchanged as they do the development.

P 1: 07110506 Ernest.rtf - 1:9 [So in these informal meetings ..] (27:27) (Super)
Codes: [Capturing Pragmatic Knowledge] [Knowledge Sharing]

So in these informal meetings people discuss the technology, discuss the process and the type of work that they're doing. And you task especially the upcoming youngsters within the organisation to actually take down the information that people are bringing. And you also ask them to access some journals and so forth, and try to relate that information to the standard practices that people are following.

P 1: 07110506 Ernest.rtf - 1:12 [So these whiteboards would be ..] (31:31) (Super)
Codes: [Discussion Forum] [Knowledge Sharing]

So these whiteboards would be - that's where people would just write those terms that they are used to, that they normally use. And then the flyers would actually - when people have a meeting they talk and so forth - then you stick those along the corridors in that organisation. So that as you walk up and down, you actually see those common terms and so forth

P 1: 07110506 Ernest.rtf - 1:38 [And besides that, when you do ..] (103:103) (Super)
Codes: [Developer Understanding of Business Model] [Knowledge Sharing]
Memos: [Studying the business environment]

And besides that, when you do such workshops, you are not just aiming at them understanding the system but you're also aiming at the developer's understanding of the business model that the organisation is using.

P 1: 07110506 Ernest.rtf - 1:58 [But what usually happens is th..] (161:161) (Super)
Codes: [Knowledge Sharing] [Language Community]
Memos: [ME - 11/13/08 [6]]

But what usually happens is that when you get to an organisation, you need to understand their language, the way they talk about things, whether they are technical or non-technical but for the purposes of your requirements. The way the information is represented will sometimes differ from other organisations even in the same field.

P 8: 08072200 and 08072201-Sindiso.rtf - 8:45 [So it is crucial that as a tea..] (136:136) (Super)
Codes: [Knowledge Sharing]

Code: Lack of humanist understanding {1-0}

P 5: 07121401-Barry Myburgh Corrected.rtf - 5:9 [And therefore I suppose I do a..] (27:27) (Super)
Codes: [Challenge for Software development] [Lack of humanist
understanding]

And therefore I suppose I do agree that they're not programming them correctly but I'm not sure that it is possible to get that point where the computer can be as capable of understanding human speech the way that human beings can understand it. I don't necessarily think that it's an achievable goal.

Code: Language Community {7-0}~

P 1: 07110506 Ernest.rtf - 1:14 [So instead of people being stu..] (33:33) (Super)
Codes: [Concept negotiation Technique] [Language Community]
Memos: [ME - 11/12/08 [1]]

So instead of people being stuck to those technical terms and so forth, they come up with some naming convention that is common to everyone. And that naming convention brings everyone to the same language, despite the differences and so forth.

P 1: 07110506 Ernest.rtf - 1:28 [So sitting together there is a..] (67:67)   (Super)
Codes: [Language Community] [User Involvement]

So sitting together there is a very important aspect or practice of eliciting the requirements or systems development. You are saying that these three guys should be together. You don't want the systems analyst to meet some guys there, get some documents signed and then go and shift that information into the programme, but you want these three levels to actually sit together. The user must be the centre of all those things.

P 1: 07110506 Ernest.rtf - 1:58 [But what usually happens is th..] (161:161)   (Super)
Codes: [Knowledge Sharing] [Language Community]
Memos: [ME - 11/13/08 [6]]

But what usually happens is that when you get to an organisation, you need to understand their language, the way they talk about things, whether they are technical or non-technical but for the purposes of your requirements. The way the information is represented will sometimes differ from other organisations even in the same field.

P 3: 07111501 Magda.rtf - 3:14 [And there should be users at o..] (72:72)   (Super)
Codes: [Language Community]

And there should be users at our side when we develop these requirements, and we should give them examples.

P 5: 07121401-Barry Myburgh Corrected.rtf - 5:15 [in fact one of my customers a ..] (53:53)   (Super)
Codes: [Language Community] [Need for a communication medium.] [Reality
Centre]

in fact one of my customers a few years back, they had a centre - a room which was dedicated to sharing information, they called it a reality centre. And that sort-of supports my thinking as well, where the best way to communicate is to use a data projector, you use some kind of tool. I'm a great believer in using mind-mapping as a tool to share thoughts and to achieve consensus. So I would typically project on the screen as the mind-map develops, as I'm engaging with the people that have to be part of the buy-in in the model that is developed, and they participate in the development of the model. So it's very much like in a real JAD session, where you use some tools to develop a model and you involve everyone in visually in this kind of reality centre concept of participating in the development of the model.

P 5: 07121401-Barry Myburgh Corrected.rtf - 5:26 [I tried to be sensitive to the..] (99:99)   (Super)
Codes: [Language Community]

I tried to be sensitive to the fact that the words people use don't necessarily mean the same thing to different people. So I'm always on the lookout for that possibility and then just use effective listening techniques - you know, reflection, confirmation of understanding and so on, to try to ensure that we have actually got the same meaning although we started out using different words.

P 8: 08072200 and 08072201-Sindiso.rtf - 8:44 [So it is quite important that,..] (134:134)   (Super)
Codes: [Language Community]

Code: language database {1-0}~

P 1: 07110506 Ernest.rtf - 1:59 [language database] (163:163)   (Super)
Codes: [language database]

language database

Code: Language Limitations {2-0}~

P 1: 07110506 Ernest.rtf - 1:56 [For example, when somebody exp..] (155:155)   (Super)
Codes: [Language Limitations]
Memos: [ME - 11/13/08 [4]]

For example, when somebody expresses their idea, the limitations of that expression or that representation, are within the language that they are using.

P 1: 07110506 Ernest.rtf - 1:57 [So that is exactly the demise ..] (155:155)   (Super)
Codes: [Capturing Semantics Issue] [Language Limitations]
Memos: [ME - 11/13/08 [5]]

So that is exactly the demise that we have as software developers, that if we try to go into the details of the softer side - the non-formal parts of the system - we find that we have limitations in terms of the language. Because whatever language you are using, when I try to use now the computer language to express that, it might not represent what you are talking about. So it is really the issues of language that should be dealt with and so forth.

Code: Language Requirement {4-0}

P 1: 07110506 Ernest.rtf - 1:55 [because the language is the on..] (153:153)   (Super)
Codes: [Language Requirement] [New Development Approach]
Memos: [ME - 11/13/08 [3]]

because the language is the only solid thing or it’s the only output that you get from a human being that lets you know what is happening in their brain. And if you want to deal with the informal part of the system, you are really looking at the softer issues, the human issues of software development. The behaviour of human beings and their languages would be the centre that people have to deal with in order to understand.

P 3: 07111501 Magda.rtf - 3:17 [Plain common language; not tec..] (84:84)   (Super)
Codes: [Language Requirement]

Plain common language; not technical jargon. Sometimes we don’t even - we are not familiar with our own language, the terms we use, the slang that we use. So we should really, really use just plain common language at the level that the user can understand.

P 5: 07121401-Barry Myburgh Corrected.rtf - 5:27 [the reality centre and modelli..] (101:101)   (Super)
Codes: [Concept negotiation Technique] [Language Requirement]

the reality centre and modelling things with everybody busily participating in the modelling that takes place, that to me is a very effective way of doing that mediation. Because if you are busy developing something like a mind map, you know, you type in the words that you think describe the point that has been made and the people who had made the point immediately see the words you use.

P 8: 08072200 and 08072201-Sindiso.rtf - 8:5 [But like I said, something bet..] (34:34)   (Super)
Codes: [Language Requirement]

Code: last partitioning {1-0}

P 9: 08081800 Kabelo.rtf - 9:10 [last partitioning] (9:9)   (Super)
Codes: [last partitioning]

Code: Link amongst stakeholders in software development {2-0}

P 9: 08081800 Kabelo.rtf - 9:45 [It goes with the methodology, ..] (109:109)   (Super)
Codes: [Link amongst stakeholders in software development]

P 9: 08081800 Kabelo.rtf - 9:46 [So, it goes with the methodolo..] (113:113)   (Super)
Codes: [Link amongst stakeholders in software development] [Need for
concept negotiation method]

Code: Link specifications to programming language {1-0}

P 8: 08072200 and 08072201-Sindiso.rtf - 8:38 [I personally feel, it's a good..] (122:122)   (Super)
Codes: [Link specifications to programming language]

Code: load testing {1-0}

P 8: 08072200 and 08072201-Sindiso.rtf - 8:54 [load testing] (148:148)   (Super)
Codes: [load testing]

Code: Managing Scope Creep {3-0}~

P 5: 07121401-Barry Myburgh Corrected.rtf - 5:38 [In a controlled quality enviro..] (125:125)   (Super)
Codes: [Managing Scope Creep]

In a controlled quality environment I apply change control. I go through all the mechanisms of managing change requests, evaluating the impact of the change, getting the customer to either agree to postpone the change, or pay more money for it, or to take more time for it or whatever. So change control is the answer.

P 5: 07121401-Barry Myburgh Corrected.rtf - 5:39 [In an evolutionary adaptive - ..] (127:127)   (Super)
Codes: [Managing Scope Creep]

In an evolutionary adaptive - no what - a crafted quality, agile world I probably manage it quite differently. I say to myself - and that's where the customer collaboration is so important because the customer must be a part of that decision to say "okay this change that we are now considering, is it more important than the objectives that we set for ourselves at the beginning of this timebox. And if we agree it is more important, then we add that to the scope of work and we'll probably accept at the same time that some of the other work that was intended will fall off the slate because there's just not time and resources to do it".

P 7: 08072102 Edwin.rtf - 7:11 [And one of the best things tha..] (65:65)   (Super)
Codes: [Managing Scope Creep]

Code: Mechanistic View {0-2}

Code: meeting sessions {1-0}

P 8: 08072200 and 08072201-Sindiso.rtf - 8:22 [meeting sessions] (80:80)   (Super)
Codes: [meeting sessions]

Code: Microsoft SharePoint {1-0}

P 9: 08081800 Kabelo.rtf - 9:38 [In Nedbank they were using Mic..] (93:93)   (Super)

Codes: [Communication Technique] [Microsoft SharePoint]

Code: Mind Mapping {1-0}

P 9: 08081800 Kabelo.rtf - 9:36 [Else we'll use your mind maps ..] (87:87)   (Super)
Codes: [Mind Mapping]

Code: Need a Variety of Development Tools {2-0}

P 1: 07110506 Ernest.rtf - 1:36 [that you are dealing with peop..] (97:97)   (Super)
Codes: [Need a Variety of Development Tools]
Memos: [ME - 11/13/08]

that you are dealing with people and people are important in those systems, and what they know is also important. It means that if you try to come up with tools like those, you are threatening the development approach itself. The methodology is threatened because you are restricting to it a specific tool within the project. Remember the projects are different and the skill sets of people that you work with are different.

P 8: 08072200 and 08072201-Sindiso.rtf - 8:12 [I agree, it's basically the po..] (66:66)   (Super)
Codes: [Need a Variety of Development Tools] [Need for a communication
medium.] [Problems in Software development]

Code: Need for a communication medium. {3-0}~

P 1: 07110506 Ernest.rtf - 1:30 [One group of clients were clie..] (73:73)   (Super)
Codes: [Need for a communication medium.]
Memos: [ME - 11/12/08 [8]]

One group of clients were clients who really don't know much about systems development.

P 5: 07121401-Barry Myburgh Corrected.rtf - 5:15 [in fact one of my customers a ..] (53:53)   (Super)
Codes: [Language Community] [Need for a communication medium.] [Reality
Centre]

in fact one of my customers a few years back, they had a centre - a room which was dedicated to sharing information, they called it a reality centre. And that sort-of supports my thinking as well, where the best way to communicate is to use a data projector, you use some kind of tool. I'm a great believer in using mind-mapping as a tool to share thoughts and to achieve consensus. So I would typically project on the screen as the mind-map develops, as I'm engaging with the people that have to be part of the buy-in in the model that is developed, and they participate in the development of the model. So it's very much like in a real JAD session, where you use some tools to develop a model and you involve everyone in visually in this kind of reality centre concept of participating in the development of the model.

P 8: 08072200 and 08072201-Sindiso.rtf - 8:12 [I agree, it's basically the po..] (66:66)   (Super)
Codes: [Need a Variety of Development Tools] [Need for a communication
medium.] [Problems in Software development]

Code: Need for a development tool {2-0}

P 2: 07111500 Goede.rtf - 2:24 [Firstly because I think it's m..] (75:75)   (Super)
Codes: [Need for a development tool] [Requirements Repository]
Memos: [Requirements Repository]

Firstly because I think it's mostly - let me say, I don't think the end-user will benefit as much as the developer from doing it because you have one version of the truth of the requirement, you have one place - central repository -

where you can find the requirements, you can set up checklists to see if you have fulfilled the requirements, the requirements are (nearby??) but then the system must force the developers or the analysts to fill it in.

P 8: 08072200 and 08072201-Sindiso.rtf - 8:7 [So I mean the process of migra..] (40:40) (Super)
Codes: [Need for a development tool]

Code: Need for a language between analysis and design models {1-0}

P 8: 08072200 and 08072201-Sindiso.rtf - 8:9 [The conceptual view, usually i..] (57:57) (Super)
Codes: [Need for a language between analysis and design models]

Code: Need for concept negotiation method {3-0}

P 8: 08072200 and 08072201-Sindiso.rtf - 8:43 [“by interface, are you relatin..] (132:132) (Super)
Codes: [Need for concept negotiation method]

P 8: 08072200 and 08072201-Sindiso.rtf - 8:46 [At the bank we have this appli..] (138:138) (Super)
Codes: [Need for concept negotiation method]

P 9: 08081800 Kabelo.rtf - 9:46 [So, it goes with the methodolo..] (113:113) (Super)
Codes: [Link amongst stakeholders in software development] [Need for
concept negotiation method]

Code: Need for documentation {1-0}~

P 8: 08072200 and 08072201-Sindiso.rtf - 8:34 [So in essence, our document, a..] (102:102) (Super)
Codes: [Need for documentation]

Code: Need to adapt systems to organizational requirements {1-0}

P 8: 08072200 and 08072201-Sindiso.rtf - 8:8 [You can have a different - say..] (46:47) (Super)
Codes: [Need to adapt systems to organizational requirements]

Code: Need to improve programming languages {5-0}~

P 8: 08072200 and 08072201-Sindiso.rtf - 8:1 [In essence, like you’re saying..] (23:23) (Super)
Codes: [Need to improve programming languages]

P 8: 08072200 and 08072201-Sindiso.rtf - 8:2 [So in a way it’s in the struct..] (25:25) (Super)
Codes: [Need to improve programming languages]

P 8: 08072200 and 08072201-Sindiso.rtf - 8:24 [So in a essence we need to get..] (86:86) (Super)
Codes: [Need to improve programming languages]

P 8: 08072200 and 08072201-Sindiso.rtf - 8:25 [first of all I believe we shou..] (88:88) (Super)
Codes: [Need to improve programming languages] [Requirements of a new
development approach]

P 8: 08072200 and 08072201-Sindiso.rtf - 8:27 [So, first of all we need to de..] (90:90) (Super)
Codes: [Need to improve programming languages]

Code: Negating the analysis model characteristics {1-0}

P 8: 08072200 and 08072201-Sindiso.rtf - 8:11 [So, in essence, they tend to f..] (61:61)   (Super)
Codes: [Negating the analysis model characteristics]

Code: New Development Approach {3-0}

P 1: 07110506 Ernest.rtf - 1:17 [So I think the best way to act..] (43:43)   (Super)
Codes: [New Development Approach]
Memos: [ME - 11/12/08 [3]]

So I think the best way to actually represent that information will be to come up with some - not a framework - but maybe language of some sort that has the ability to actually express those concepts and to relate to technical issues that way. So I'm not very sure of what language you could actually use for that but I'm tending to think in terms of the approaches to development.

P 1: 07110506 Ernest.rtf - 1:55 [because the language is the on..] (153:153)   (Super)
Codes: [Language Requirement] [New Development Approach]
Memos: [ME - 11/13/08 [3]]

because the language is the only solid thing or it's the only output that you get from a human being that lets you know what is happening in their brain. And if you want to deal with the informal part of the system, you are really looking at the softer issues, the human issues of software development. The behaviour of human beings and their languages would be the centre that people have to deal with in order to understand.

P 8: 08072200 and 08072201-Sindiso.rtf - 8:13 [I would prefer, … a method in ..] (68:68)   (Super)
Codes: [Development Approach] [New Development Approach]

Code: object-oriented approach, {6-5}~

P 1: 07110506 Ernest.rtf - 1:44 [object-oriented approach,] (117:117)   (Super)
Codes: [object-oriented approach,]

object-oriented approach,

P 2: 07111500 Goede.rtf - 2:3 [Object-oriented programming to..] (19:19)   (Super)
Codes: [object-oriented approach,]
Memos: [ME - 01/12/09 [1]]

Object-oriented programming to me works well in the technical applications. Perhaps it's not yet proven enough for me in today's software development, except perhaps for requirement analysis.

P 2: 07111500 Goede.rtf - 2:4 [object-oriented technique does..] (23:23)   (Super)
Codes: [object-oriented approach,]

object-oriented technique does allow more for people to capture all their requirements, to capture non-technical things.

P 3: 07111501 Magda.rtf - 3:4 [object-orientated paradigm] (27:27)   (Super)
Codes: [object-oriented approach,]

object-orientated paradigm

P 3: 07111501 Magda.rtf - 3:37 [Object-orientation, I think, i..] (232:232)   (Super)
Codes: [Capturing Semantics Issue] [object-oriented approach,]

Object-orientation, I think, if you look at the instantiation of a class, then I would say that could - object would be the - ja, the semantics behind the design. You design the class and then an instantiation of the class, the object would be the semantics.

P 8: 08072200 and 08072201-Sindiso.rtf - 8:3 [The object oriented approach s..] (28:28)   (Super)
Codes: [object-oriented approach,]

Code: object-oriented paradigm {1-0}

P 5: 07121401-Barry Myburgh Corrected.rtf - 5:4 [in principle the object-orient..] (17:17)   (Super)
Codes: [object-oriented paradigm]

in principle the object-oriented paradigm lends itself to modelling reality; and from that perspective then I think, you know, in principle it should be better positioned to perhaps then capture those semantics as you described them.

Code: Object orientation versus structured {1-0}

P 3: 07111501 Magda.rtf - 3:11 [The object orientation, if you..] (57:57)   (Super)
Codes: [Object orientation versus structured]

The object orientation, if you look at the use case diagrams, and you get the documentation of all the use cases, okay that could be more meaningful in a sense. There could be some semantics in there. Okay, I would say both but I would say the object orientation really could be more meaningful. I

Code: object oriented language {1-0}

P 8: 08072200 and 08072201-Sindiso.rtf - 8:30 [object oriented language] (96:96)   (Super)
Codes: [object oriented language]

Code: Ontology Knowledge Base {1-0}

P 1: 07110506 Ernest.rtf - 1:61 [So what I'm saying is that you..] (165:165)   (Super)
Codes: [Ontology Knowledge Base] [Process Ontology]
Memos: [ME - 11/13/08 [8]]

So what I'm saying is that you need some formal way where you can have a database that has some schemas of those words, of those concepts that we're talking about. For example, if we talk about a loan, all the financial institutions will have something called a loan. So you can have a loan schema, you find that your Nedbank, your ABSAs and Standard Banks might actually have different terms for your interest - for the interest rate, for whatever, for the borrowing rate or whatever - you know, those technical terms that they use. They might have different terms that actually mean the same thing, so your schema must be able to capture that - ensure that it is the same thing so that when you now check back and re-use it in different institutions you are not really changing much. It will come out there according to their language but you are still using the same thing
----

Code: Oracle Designer {1-1}~

P 3: 07111501 Magda.rtf - 3:24 [Oracle Designer] (148:148)   (Super)
Codes: [Oracle Designer]

Oracle Designer

Code: Pair Programming {1-2}~

P 1: 07110506 Ernest.rtf - 1:6 [Pair Programming] (25:25)   (Super)
Codes: [Pair Programming]

Pair Programming

Code: peer reviews. {2-0}

P 9: 08081800 Kabelo.rtf - 9:27 [peer reviews.] (53:53)   (Super)
Codes: [peer reviews.]

P 9: 08081800 Kabelo.rtf - 9:54 [Peer reviews you will see them..] (171:171)   (Super)
Codes: [peer reviews.]

Code: Popular methodologies in Banking industry {1-0}

P 9: 08081800 Kabelo.rtf - 9:22 [But with the fact the - my pre..] (29:29)   (Super)
Codes: [Popular methodologies in Banking industry]

Code: Problems with case tools {1-0}~

P 3: 07111501 Magda.rtf - 3:25 [So it could help. In my experi..] (150:150)   (Super)
Codes: [Problems with case tools]
Memos: [Case tools problem]

So it could help. In my experience through my research, I discovered that sometimes the analysis takes longer than what the people or the developers were used to and then they get discouraged. They should just keep with the process, although it takes longer. In the end you save a lot of time because this is generated automatically (inaudible - noise). My students use this for the last three years - they used Oracle Designer. I saw that with them also, they felt “og, we are busy - we are just busy with the analysis and design, we can’t make progress”. Then all of a sudden you’ll generate the code and then they see.

Code: Pragmatics {0-3}

Code: Problem of Communication {1-0}

P 5: 07121401-Barry Myburgh Corrected.rtf - 5:8 [people operate and communicate..] (27:27)   (Super)
Codes: [Problem of Communication]

people operate and communicate very often with a great measure of vagueness and imprecision, and they use words that mean something to one person, that means something different to another person. And that seems to be the reality of how human beings communicate with each other. To expect a computer to be able to cope with that kind of lack of precision in the way that people speak, and the words they use, and the meanings that they attach to those words, I actually think - I don’t quite agree with this.

Code: Problem of reusing requirements across organizations {2-0}

P 8: 08072200 and 08072201-Sindiso.rtf - 8:28 [Requirements on their own …are..] (94:94) (Super)
Codes: [Problem of reusing requirements across organizations]

P 8: 08072200 and 08072201-Sindiso.rtf - 8:29 [Or you could find the frame, t..] (94:94) (Super)
Codes: [Problem of reusing requirements across organizations]

Code: Problem with design model {1-0}~

P 8: 08072200 and 08072201-Sindiso.rtf - 8:10 [So, in essence it bypasses a l..] (59:59) (Super)
Codes: [Problem with design model]

Code: Problem with use of Object Oriented Paradigm {1-0}

P 5: 07121401-Barry Myburgh Corrected.rtf - 5:6 [My problem as I see in industr..] (19:19) (Super)
Codes: [Problem with use of Object Oriented Paradigm]

My problem as I see in industry where I operate, people call things object-oriented when in fact it’s just another name for structured modular implementation. And I therefore, you know, I think there’s a great lack of - in the broader IT space, there’s a great lack of effective utilisation of the object-oriented paradigm.

Code: Problems in Software development {3-0}

P 5: 07121401-Barry Myburgh Corrected.rtf - 5:18 [But of course the collaboratio..] (59:59) (Super)
Codes: [Problems in Software development]

But of course the collaboration is easier said than done because people are not available. The client is running his or her business and that’s their first priority and they can’t suddenly turn off that part of their life to join your project team and be with you all the time. So it’s really a very difficult thing to achieve.

P 5: 07121401-Barry Myburgh Corrected.rtf - 5:28 [The problem with it of course ..] (103:103) (Super)
Codes: [Problems in Software development]

The problem with it of course is it takes time. That’s one of the things that we often don’t have and therefore we don’t always get it right.

P 8: 08072200 and 08072201-Sindiso.rtf - 8:12 [I agree, it’s basically the po..] (66:66) (Super)
Codes: [Need a Variety of Development Tools] [Need for a communication
medium.] [Problems in Software development]

Code: Problems of not involving the users {1-0}

P 9: 08081800 Kabelo.rtf - 9:44 [But now you find out that he m..] (105:105) (Super)
Codes: [Communication Problem] [Problems of not involving the users]

Code: Problems of using Case tool {1-0}

P 5: 07121401-Barry Myburgh Corrected.rtf - 5:20 [I’ve got the unfulfilled hope ..] (71:71) (Super)
Codes: [Problems of using Case tool]

I’ve got the unfulfilled hope that it would really achieve great productivity gains and so on. Unfortunately, when customers are in the kind of craziness of the business reality, where they think that speed is all that counts, then my experience is that case tools don’t necessarily help so much. In fact, in some situations the case tools have been seen to be a hindrance because the case tool effectively forces the people doing the work to be more thorough and to cover all the aspects of the modelling and sophistication of the solution. And there just isn't time to do all of that so people find the tool becomes a problem; they can't use it effectively. And people tend to discard - they end up discarding the use of the tools and really just go through to the minimum level of sort-of version control tools or something like that.

Code: process-orientated paradigm, {1-2}

P 3: 07111501 Magda.rtf - 3:1 [In the process-orientated para..] (25:25)   (Super)
Codes: [process-orientated paradigm,]

In the process-orientated paradigm, the philosophical view of the developers of this methodologies is that the process is the most important aspect of the system
----

Code: process chart {1-2}~

P 2: 07111500 Goede.rtf - 2:11 [process chart] (51:51)   (Super)
Codes: [process chart]

process chart

Code: Process Ontology {1-0}

P 1: 07110506 Ernest.rtf - 1:61 [So what I'm saying is that you..] (165:165)   (Super)
Codes: [Ontology Knowledge Base] [Process Ontology]
Memos: [ME - 11/13/08 [8]]

So what I'm saying is that you need some formal way where you can have a database that has some schemas of those words, of those concepts that we're talking about. For example, if we talk about a loan, all the financial institutions will have something called a loan. So you can have a loan schema, you find that your Nedbank, your ABSAs and Standard Banks might actually have different terms for your interest - for the interest rate, for whatever, for the borrowing rate or whatever - you know, those technical terms that they use. They might have different terms that actually mean the same thing, so your schema must be able to capture that - ensure that it is the same thing so that when you now check back and re-use it in different institutions you are not really changing much. It will come out there according to their language but you are still using the same thing
----

Code: Prominent development approaches {1-1}~

P 3: 07111501 Magda.rtf - 3:9 [You mentioned only two of thos..] (43:43)   (Super)
Codes: [Prominent development approaches]
Memos: [Prominent development approaches]

You mentioned only two of those approaches and in my research I saw that those two are the mostly used approaches in the industry. You find that there's a movement towards rapid application development but the structured and the object-orientated approach are indeed, I think, the most used approaches.

Code: proof-of-concept {3-1}~

P 1: 07110506 Ernest.rtf - 1:32 [proof-of-concept which was mor..] (79:79)   (Super)
Codes: [proof-of-concept]

proof-of-concept which was more practical, where you're coming up with a demo of the system, how the system is going to work. That's basically what really impressed them because they could now start contributing and saying "okay, at this point we want this" or "at this point, no, we don't deliver this kind of reports.

P 2: 07111500 Goede.rtf - 2:12 [A prototype tells you sometime..] (51:51)   (Super)
Codes: [proof-of-concept]

A prototype tells you sometimes that - it's empty behind that screen and the proof-of-concept has some functionality and you should have a bit of functionality.

P 2: 07111500 Goede.rtf - 2:19 [that is to do a very good veri..] (63:63)   (Super)
Codes: [proof-of-concept]

that is to do a very good verification after the requirements has been gathered. That can be done, once again, with the proof-of-concept.

Code: prototype {2-3}

P 2: 07111500 Goede.rtf - 2:13 [A prototype tells you sometime..] (51:51)   (Super)
Codes: [prototype]

A prototype tells you sometimes that - it's empty behind that screen and the proof-of-concept has some functionality and you should have a bit of functionality.

P 3: 07111501 Magda.rtf - 3:15 [You know, we can talk and peop..] (72:72)   (Super)
Codes: [prototype]

You know, we can talk and people can come and they can tell you stuff, and then I would use prototypes. And then just very, the prototypes that you can throw away after you've done your requirements. If you re-use your prototypes, that that prototype might become the system and then you don't focus on the requirements. So I would have - my user and I would use prototyping all the way, just to show the user "is this what you want".

Code: Purpose of analysis stage {1-0}

P 9: 08081800 Kabelo.rtf - 9:53 [then we have to move into the ..] (169:169)   (Super)
Codes: [Purpose of analysis stage]

Code: quality gate {2-0}~

P 9: 08081800 Kabelo.rtf - 9:30 [with regard to the V model the..] (59:59)   (Super)
Codes: [quality gate]

P 9: 08081800 Kabelo.rtf - 9:34 [If ever - those requirements d..] (83:83)   (Super)
Codes: [quality gate] [User Involvement]

Code: Quality Measure {1-0}

P 1: 07110506 Ernest.rtf - 1:40 [So the client must be there th..] (107:107)   (Super)
Codes: [Quality Measure] [User Involvement]
Memos: [ME - 11/13/08 [1]]

So the client must be there throughout and that is basically for quality check purposes, to ensure that the kind of

requirements that the user has are actually met.

Code: rapid application development... {1-1}~

P 3: 07111501 Magda.rtf - 3:8 [rapid application development...] (39:39) (Super)
Codes: [rapid application development...]

rapid application development.

Code: realistic worldview {1-3}~

P 2: 07111500 Goede.rtf - 2:8 [The realistic worldview says t..] (43:43) (Super)
Codes: [realistic worldview]

The realistic worldview says that everything is agreeable. You and I can decide this is a mouse and this is the function of the mouse. That is the realistic worldview and that is where the problem with the engineers comes in. They follow a realistic worldview. They think they agree on the purpose of this thing.

Code: Reality Centre {1-0}

P 5: 07121401-Barry Myburgh Corrected.rtf - 5:15 [in fact one of my customers a ..] (53:53) (Super)
Codes: [Language Community] [Need for a communication medium.] [Reality
Centre]

in fact one of my customers a few years back, they had a centre - a room which was dedicated to sharing information, they called it a reality centre. And that sort-of supports my thinking as well, where the best way to communicate is to use a data projector, you use some kind of tool. I'm a great believer in using mind-mapping as a tool to share thoughts and to achieve consensus. So I would typically project on the screen as the mind-map develops, as I'm engaging with the people that have to be part of the buy-in in the model that is developed, and they participate in the development of the model. So it's very much like in a real JAD session, where you use some tools to develop a model and you involve everyone in visually in this kind of reality centre concept of participating in the development of the model.

Code: regressional testing {1-0}

P 8: 08072200 and 08072201-Sindiso.rtf - 8:52 [regressional testing] (148:148) (Super)
Codes: [regressional testing]

Code: relativistic worldview {1-2}~

P 2: 07111500 Goede.rtf - 2:9 [And if we follow a relativisti..] (45:45) (Super)
Codes: [relativistic worldview]

And if we follow a relativistic worldview, you might have an opinion on this device and I can have a different opinion on this device. And if we then get down to a discussion, we might design a much better mouse than this - if we sit down and think "what is the real purpose".

Code: Requirement for non prescriptive ways of using systems {1-0}~

P 5: 07121401-Barry Myburgh Corrected.rtf - 5:13 [So to my mind, that is the way..] (47:47) (Super)

Codes: [Requirement for non prescriptive ways of using systems]

So to my mind, that is the way that things should be done, that the system is built from the perspective of trying to anticipate any - you know, a whole set of ways in which a user would prefer to actually perform a certain function; and then leave it to the user to choose the route that is most comfortable for him or her.

Code: Requirements for quality software products {1-0}

P 5: 07121401-Barry Myburgh Corrected.rtf - 5:30 [First of all, you have to have..] (111:111)   (Super)
Codes: [Requirements for quality software products]

First of all, you have to have enough time to drive out the proper understanding of the requirements of the architecture, and to plan the project against a sufficiently understood reality. If you're in a great hurry and there's no time to do planning, there's no time to develop an architecture and things like that, then you will not get those things right.

Code: Requirements of a Development Tool {4-1}~

P 1: 07110506 Ernest.rtf - 1:33 [Looking at it from that point ..] (89:89)   (Super)
Codes: [Requirements of a Development Tool]

Looking at it from that point of view, I would say that the most important parts of a tool that I would expect is that tool is going to be used in reusable systems and adaptable systems and so forth, it should be a tool that is open enough to allow the users to bring in their expertise and skills.

P 1: 07110506 Ernest.rtf - 1:35 [The other thing that I would a..] (91:91)   (Super)
Codes: [Requirements of a Development Tool]
Memos: [ME - 11/12/08 [9]]

The other thing that I would also mention about the competence of the tool, is that the tool should not be too much restricted to a particular methodology because, like you said, a methodology can actually change, people can come up with different approaches to that methodology, and change the framework and its parameters. So when that happens, the tool itself must actually be adaptable to that.

P 8: 08072200 and 08072201-Sindiso.rtf - 8:35 [You know, we have a system tha..] (108:108)   (Super)
Codes: [Requirements of a Development Tool]

P 8: 08072200 and 08072201-Sindiso.rtf - 8:36 [I mean the system on its own s..] (110:110)   (Super)
Codes: [Requirements of a Development Tool] [Requirements of a new
development approach]

Code: Requirements of a new development approach {4-0}

P 8: 08072200 and 08072201-Sindiso.rtf - 8:14 [You have a computer system whi..] (68:68)   (Super)
Codes: [Requirements of a new development approach]

P 8: 08072200 and 08072201-Sindiso.rtf - 8:25 [first of all I believe we shou..] (88:88)   (Super)
Codes: [Need to improve programming languages] [Requirements of a new
development approach]

P 8: 08072200 and 08072201-Sindiso.rtf - 8:36 [I mean the system on its own s..] (110:110)   (Super)
Codes: [Requirements of a Development Tool] [Requirements of a new
development approach]

P 8: 08072200 and 08072201-Sindiso.rtf - 8:37 [So, like I said, it's basicall..] (112:112) (Super)
Codes: [Adaptive systems development approaches] [Requirements of a new development approach] [Romantic Worldview]

Code: Requirements of a software tester {1-0}

P 9: 08081800 Kabelo.rtf - 9:40 [I mean as the software tester,..] (101:101) (Super)
Codes: [Requirements of a software tester]

Code: Requirements Repository {2-2}~

P 2: 07111500 Goede.rtf - 2:24 [Firstly because I think it's m..] (75:75) (Super)
Codes: [Need for a development tool] [Requirements Repository]
Memos: [Requirements Repository]

Firstly because I think it's mostly - let me say, I don't think the end-user will benefit as much as the developer from doing it because you have one version of the truth of the requirement, you have one place - central repository - where you can find the requirements, you can set up checklists to see if you have fulfilled the requirements, the requirements are (nearby??) but then the system must force the developers or the analysts to fill it in.

P 8: 08072200 and 08072201-Sindiso.rtf - 8:33 [So basically I would document ..] (102:102) (Super)
Codes: [Requirements Repository]

Code: Reusability of Requirements {3-0}

P 5: 07121401-Barry Myburgh Corrected.rtf - 5:24 [So I suppose the answer in pri..] (91:91) (Super)
Codes: [Reusability of Requirements]

So I suppose the answer in principle is to ensure that requirements can be re-used, you need to have a sufficiently organised approach to your software engineering so that you can have access to the designs and specifications and architecture of systems that have previously been built. It's to do with configuration management and if your discipline of configuration management isn't effectively implemented, then you just can't achieve that.

P 8: 08072200 and 08072201-Sindiso.rtf - 8:31 [object oriented language is ac..] (96:96) (Super)
Codes: [Reusability of Requirements]

P 8: 08072200 and 08072201-Sindiso.rtf - 8:32 [So, in essence you create obje..] (98:98) (Super)
Codes: [Reusability of Requirements]

Code: Romantic Worldview {1-8}

P 8: 08072200 and 08072201-Sindiso.rtf - 8:37 [So, like I said, it's basicall..] (112:112) (Super)
Codes: [Adaptive systems development approaches] [Requirements of a new development approach] [Romantic Worldview]

Code: schemas {1-0}

P 1: 07110506 Ernest.rtf - 1:60 [So now what I'm trying to say ..] (165:165) (Super)
Codes: [schemas]

So now what I'm trying to say is that you need to come up with some schemas because that language is full of these schemas that you used to, for example, you're trying to come up with a system for the library. In order for you to represent the requirements there, you come up with those schemas that will give you, for example, the - you can

have schema called “loaning a book” or something like that.

Code: scope creep {2-2}~

P 2: 07111500 Goede.rtf - 2:31 [I think scope creep always hap..] (103:103) (Super)
Codes: [scope creep]

I think scope creep always happens but the extent of it depends on your - two things - actually mainly on the quality of your requirements. If your requirements are done well and you involve the right people, you might have a situation where you don’t get that much scope through it. And then as - if you are using a methodology where your end-user is involved, then scope should be their problem, not your problem in a sense. But that’s only the case if you have a end-user or a - I don’t want to use the word end-user, I want to use a word (indiscernible - audio quality) user in data warehousing, business sponsors - somebody higher up in the organisation. Somebody more important than the secretary that’s going to type on the system; somebody that has power inside the organisation. I want to have that person involved and he should prioritise all these additional scope requirements, not the programmers.

P 3: 07111501 Magda.rtf - 3:35 [Is spent on 20 percent of the ..] (214:216) (Super)
Codes: [scope creep]
Memos: [Managing scope creep]

Is spent on 20 percent of the requirement. Are they willing to do that? Are they willing to spend 80 percent of the time? If they cut that 20 percent of the requirement, they could save 80 percent of the time you can go onto a next project. So just discuss it with them. Ask them what their philosophy is, I would say, because some people really want that product and they want it in the way they visualise it and they want to add stuff and so on. And if they are willing to pay for that, fine with me, then I will add it because in the end of the day, it’s [them] that are paying for the product, not me.

But other companies are very, very, very driven and they want you to produce quick results and then I would show them. Think about it, if you are in your class also, the amount of time you spend with students in your office. It could be one student that takes up all your time in the office. It’s the same with this principle. There’s a small amount of requirement and it takes 80 percent of your time.

Code: Seeking explanation of concepts {3-0}

P 5: 07121401-Barry Myburgh Corrected.rtf - 5:5 [Maybe you should just explain ..] (13:13) (Super)
Codes: [Seeking explanation of concepts]

Maybe you should just explain a little bit what you mean by your use of the term semantics.

P 5: 07121401-Barry Myburgh Corrected.rtf - 5:7 [What do we mean by natural lan..] (23:23) (Super)
Codes: [Seeking explanantion of concepts]

What do we mean by natural language speakers? I mean, what are we actually talking about there? Do we expect - you know, I don’t quite understand the context of this comment. Do you mean in terms of the way that we give them instructions, that we should just be able to communicate our instructions to the computer in natural language? Is that what you mean?

P 5: 07121401-Barry Myburgh Corrected.rtf - 5:10 [You see, now I first have to c..] (31:31) (Super)
Codes: [Seeking explanantion of concepts]

You see, now I first have to check the meaning of these words now. Formal, informal and the technical part, are you

meaning by the technical part, the hardware, the actual physical -
----

Code: Semantics {0-1}

Code: So you need to try to anticipa.. {1-0}~

P 5: 07121401-Barry Myburgh Corrected.rtf - 5:14 [So you need to try to anticipa..] (49:49)   (Super)
Codes: [So you need to try to anticipa..]

So you need to try to anticipate all the different ways that people might want to perform a function and build a system to cater for them.

Code: Soft issues {2-2}~

P 1: 07110506 Ernest.rtf - 1:16 [soft issues] (41:41)   (Super)
Codes: [Soft issues]

soft issues

P 1: 07110506 Ernest.rtf - 1:52 [I would say [that] if an indiv..] (141:141)   (Super)
Codes: [Soft issues] [Soft Systems Metholodogy]

I would say [that] if an individual wants to work in that area, they'll have to look into a lot of other areas that may not necessarily be in IT but others are IT. For example going into psychology, going into organisational behaviour. Those are areas that are very, very interesting and there are books that people have written that bring the softer issues of IT and the technical issues together and so forth.

Code: Soft Systems Methodology {2-3}

P 1: 07110506 Ernest.rtf - 1:52 [I would say [that] if an indiv..] (141:141)   (Super)
Codes: [Soft issues] [Soft Systems Methodology]

I would say [that] if an individual wants to work in that area, they'll have to look into a lot of other areas that may not necessarily be in IT but others are IT. For example going into psychology, going into organisational behaviour. Those are areas that are very, very interesting and there are books that people have written that bring the softer issues of IT and the technical issues together and so forth.

P 3: 07111501 Magda.rtf - 3:6 [The other paradigm would be - ..] (29:29)   (Super)
Codes: [Soft Systems Methodology]

The other paradigm would be - I would call holistic paradigm where they focus on the system - the systems thinking. I don't know if you are familiar with the term systems thinking?

Code: Software Crisis {1-0}

P 5: 07121401-Barry Myburgh Corrected.rtf - 5:29 [The bottom line is that invari..] (111:111)   (Super)
Codes: [Software Crisis]

The bottom line is that invariably we don't ensure those things. Okay, we most often don't get it right. So if you are in a situation where you have to ensure that all of those things are achieved, well there's a few key things that you

have to be able to do.

Code: software development metaphor {1-2}~

P 1: 07110506 Ernest.rtf - 1:13 [software development metaphor] (33:33)   (Super)
Codes: [software development metaphor]

software development metaphor

Code: Software development methodology requirements {1-1}

P 2: 07111500 Goede.rtf - 2:32 [I think again, once again the ..] (109:109)   (Super)
Codes: [Software development methodology requirements]

I think again, once again the human factor is much more important than previously. We should allow for methodologies to really get us to ask the right questions for the people on how they use the system, and what their expectations are. And we should go from a relativistic viewpoint, that we understand that different people will have different views on these things. The more people you ask the better your understanding. The more different answers you get to the same question, you will enrich understanding and not hinder your understanding.

Code: software engineering environme.. {2-2}~

P 3: 07111501 Magda.rtf - 3:23 [There are those types of tools..] (148:148)   (Super)
Codes: [software engineering environme..]

There are those types of tools. Do you call it a tool? I would call it - sometimes I call it a software engineering environment. Just to make the difference clear, a tool - I would say a tool would be something that helps you draw a dataflow diagram. But what you described there I would call a software engineering environment. Ja, for example, one example of those is Oracle Designer. I don't know if you know the thing. What you do there, you start with your analysis, you go through to your design, and you do both analysis and design of processes and of your data. You mention the data there. Then in the end, you just generate - automatically generate all the code. If you do it correctly, it's fabulous, it could work wonders but there's a very big danger, you should be careful with your analysis. If you do your analysis wrong, then you can solve the problem but it's the wrong answer to the initial problem

P 5: 07121401-Barry Myburgh Corrected.rtf - 5:19 [If the customer that you are b..] (71:71)   (Super)
Codes: [Case Tool] [software engineering environme..]

If the customer that you are busy working for is the kind of customer who understands the importance of giving you enough time to do the job properly, then the use of case tools can greatly improve the software development process.

Code: software evolution {3-2}

P 1: 07110506 Ernest.rtf - 1:47 [software evolution] (129:129)   (Super)
Codes: [software evolution]

software evolution

P 1: 07110506 Ernest.rtf - 1:48 [So his concept is that if you ..] (131:131)   (Super)
Codes: [software evolution]

So his concept is that if you go electrical engineering - a field of electrical engineering called “control engineering” - you’ll find that in order to stabilise a system, they put all systems in black boxes, especially the processes - industrial processes. They’ll put it in a black box and say “a system is an input, an output”. And then in order to stabilise that system, you are saying the output must be taped as either a part of the output or the entire output. You’ll tape it and fit it back into the input, so there must be a mixer there - some function which you represent by a summation sign to say “we are summing the input and part of the feedback”. But in so doing you are regulating the output so that the output does not spiral exponentially. Right, and that concept of spiralling exponentially you’ll find it - let’s say in speakers - if you put a mic next to the speaker, then you have negative feedback. So you are trying to control the feedback there.

P 1: 07110506 Ernest.rtf - 1:49 [So in software evolution now t..] (133:133)   (Super)
Codes: [software evolution]

So in software evolution now the Lehman concept actually says that the moment you deploy the system, and the user starts using the system, they’ll tell you some things that they don’t like about the system and whatever - things that need to be changed about the system. So that should be considered as feedback from the system that you have delivered. So take that feedback and feed it together with the input, to control the way your system should actually work. So that’s the concept. It’s trying to use that control engineering approach to actually regulate the way we develop our systems.

Code: Software Maintanability {1-0}

P 1: 07110506 Ernest.rtf - 1:46 [use software patterns and thos..] (121:121)   (Super)
Codes: [Software Maintanability]

use software patterns and those will actually assist in terms of the maintainability of the system. But object-orientation is the heart of the maintainable systems because you have a lot of information hiding and whatever, and modularisation, each class has its own internals that can not be touched by any other class. So that when you do your maintenance and checks, the methods inside a class, it doesn’t affect all the other classes
----

Code: Software migration issues and problems {1-0}~

P 8: 08072200 and 08072201-Sindiso.rtf - 8:6 [And believe me, the amount of ..] (40:40)   (Super)
Codes: [Software migration issues and problems]

Code: software patterns. {1-3}

P 1: 07110506 Ernest.rtf - 1:45 [software patterns.] (117:117)   (Super)
Codes: [software patterns.]

software patterns.

Code: software tester {1-0}

P 9: 08081800 Kabelo.rtf - 9:39 [software tester] (101:101)   (Super)
Codes: [software tester]
Code: star schemas {2-1}

P 2: 07111500 Goede.rtf - 2:15 [star schemas] (55:55)   (Super)
Codes: [star schemas]

star schemas

P 2: 07111500 Goede.rtf - 2:16 [they are the basic design. The..] (59:59)   (Super)
Codes: [star schemas]

they are the basic design. They replace the ERD. They replace the ERD but they are designed in such a way that I can teach you within three seconds how a star schema works but I won't even attempt to teach anybody an ERD.

Code: Studying the system {1-1}~

P 3: 07111501 Magda.rtf - 3:21 [It could be with experience bu..] (120:120)   (Super)
Codes: [Studying the system]

It could be with experience but I think what you mentioned here is the big danger. You shouldn't do that; you shouldn't think because it looks the same, that it is the same. You should try and figure out - there's always differences. You will know.

Code: Super users {1-0}~

P 7: 08072102 Edwin.rtf - 7:10 [Whereas normally the super-use..] (57:57)   (Super)
Codes: [Super users]

Code: System Developer {4-1}~

P 1: 07110506 Ernest.rtf - 1:24 [In the approach that I've stud..] (61:61)   (Super)
Codes: [System Developer]

In the approach that I've studied in detail and also used, you'll find that the gap between the analyst and the programmer is actually closed by coming up with somebody who's called a developer, who actually elicits the requirements directly from the user. And that user, in the methods that I'm actually talking about, becomes part of the development team which means that those details - those metaphors that I was talking about - the language that is used by the team becomes familiar to that user. And the problem - the architectural problems where the systems analyst has some design diagrams and whatever that may not be clear to the programmer ?

P 2: 07111500 Goede.rtf - 2:23 [Now I've seen a number of orga..] (71:71)   (Super)
Codes: [System Developer]
Memos: [System Developer]

Now I've seen a number of organisations employing people that has a little bit of technical skill a lot of business skills to serve as the interface between the technical people and the users.

P 8: 08072200 and 08072201-Sindiso.rtf - 8:40 [And I personally believe an an..] (124:124)   (Super)
Codes: [System Developer]

P 8: 08072200 and 08072201-Sindiso.rtf - 8:42 [So I would want a situation wh..] (126:126)   (Super)
Codes: [System Developer]

Code: System testing {3-0}~

P 8: 08072200 and 08072201-Sindiso.rtf - 8:49 [Ok, there are different method..] (148:148)   (Super)
Codes: [System testing]

P 9: 08081800 Kabelo.rtf - 9:6 [system testing] (9:9)   (Super)
Codes: [System testing]

P 9: 08081800 Kabelo.rtf - 9:41 [Meaning when I'm doing system ..] (103:103)   (Super)
Codes: [System testing]

Code: System testing concept {1-0}

P 9: 08081800 Kabelo.rtf - 9:12 [From there the analyst, meanin..] (9:9)   (Super)
Codes: [System testing concept]

Code: systems approach {1-3}~

P 2: 07111500 Goede.rtf - 2:1 [systems approach] (19:19)   (Super)
Codes: [systems approach]

systems approach

Code: tacit knowledge {1-1}~

P 1: 07110506 Ernest.rtf - 1:2 [tacit knowledge] (23:23)   (Super)
Codes: [tacit knowledge]

tacit knowledge

Code: Test Analyst duties {1-0}

P 9: 08081800 Kabelo.rtf - 9:1 [Meaning I run my test scripts ..] (3:3)   (Super)
Codes: [Test Analyst duties]

Code: test design techniques {1-0}

P 9: 08081800 Kabelo.rtf - 9:7 [test design techniques] (9:9)   (Super)
Codes: [test design techniques]

Code: Test Director {1-0}~

P 8: 08072200 and 08072201-Sindiso.rtf - 8:18 [There's a system called Test D..] (74:74)   (Super)
Codes: [Communication Method] [Test Director]

Code: Test Driven Development {1-0}

P 7: 08072102 Edwin.rtf - 7:2 [Test Driven Development] (35:35)   (Super)
Codes: [Test Driven Development]

Code: timebox approach {1-0}

P 5: 07121401-Barry Myburgh Corrected.rtf - 5:35 [In a crafted quality or agile ..] (119:119)   (Super)
Codes: [Adaptive systems development approaches] [timebox approach]

In a crafted quality or agile environment, I certainly don't make promises about things like that, and I say "give me a timebox, let's set some goals and objectives for that timebox and we will do the best we can". And at the end of the timebox we will review what we have achieved. So I can certainly ensure that the budget is adhered to because the timebox gives me a certain budget. I can achieve on time because the timebox is limited, say to six weeks, two months, three months, whatever; but I can't guarantee that I what I build will be easy to modify, fault free, and that the user will be satisfied. That we will see - on reflection see to what extent we have achieved that. If the customer says "you've achieved enough to gain my confidence" then we get another timebox to improve those things.

Code: timebox approach" {1-0}~

P 5: 07121401-Barry Myburgh Corrected.rtf - 5:34 [timebox approach"] (117:117) (Super)
Codes: [timebox approach"]

timebox approach"

Code: traditional approach {3-3}~

P 1: 07110506 Ernest.rtf - 1:23 [traditional approach, where yo..] (61:61) (Super)
Codes: [traditional approach]

traditional approach, where you have a systems analyst, you have a programmer, and then you have got the business guy, the user.

P 2: 07111500 Goede.rtf - 2:2 [I'm very much against the trad..] (19:19) (Super)
Codes: [traditional approach]
Memos: [ME - 01/12/09]

I'm very much against the traditional paradigm because I believe it leads to a case where you have a system that conforms to the requirements and not to the user's needs.

P 3: 07111501 Magda.rtf - 3:3 [In the process-orientated para..] (25:25) (Super)
Codes: [traditional approach]
Memos: [Structured development approach]

In the process-orientated paradigm, the philosophical view of the developers of this methodologies is that the process is the most important aspect of the system. And the data-orientated paradigm is where they say [that] data stays constant, so we should rather focus on the data instead of the process, it would make our lives much easier.

Code: Translation Medium Requirement {1-1}~

P 1: 07110506 Ernest.rtf - 1:22 [So you have the business that ..] (57:57) (Super)
Codes: [Development Problem] [Translation Medium Requirement]

So you have the business that side, you have got the interface which the user is using, and you have the technical details of the system itself. So those are three different levels and you find that at those three different levels the user understands the business side and the interface, but does not really understand the technical details but the developer who is the one who has the technical details. So you find that that way now, those requirements will always be presented in the way that the technical guy understands which puts the user at a disadvantage.

Code: UML {4-2}~

P 1: 07110506 Ernest.rtf - 1:18 [So it's more of the - when you..] (47:47) (Super)
Codes: [UML]
Memos: [ME - 11/12/08 [5]]

So it's more of the - when you look at an object, it gives a better understanding of the system or whatever you're dealing with but I wouldn't call it semantics.

P 1: 07110506 Ernest.rtf - 1:19 [I think it's lighter than sema..] (49:49) (Super)
Codes: [UML]

I think it's lighter than semantics. That's my bottom-line, that it doesn't give the details that semantics would give. It leaves you at a general level again and that is the advantage of object-oriented programme and the development (inaudible - noise) technology. Because with that general class you can actually inherit and start reusing the system and the like without really going into the semantic details. So I would say there's a level of understanding that has been increased, as opposed to structured programming but they still need to go deeper which would now become the semantic

P 8: 08072200 and 08072201-Sindiso.rtf - 8:4 [I mean UML it's being used as ..] (34:34) (Super)
Codes: [UML]

P 9: 08081800 Kabelo.rtf - 9:37 [Okay we use your normal UML to..] (87:87) (Super)
Codes: [UML]

Code: unit testing {1-0}

P 8: 08072200 and 08072201-Sindiso.rtf - 8:56 [unit testing] (148:148) (Super)
Codes: [unit testing]

Code: use case diagrams {2-2}~

P 2: 07111500 Goede.rtf - 2:5 [use case diagrams] (23:23) (Super)
Codes: [use case diagrams]
Memos: [ME - 01/12/09 [2]]

use case diagrams

P 2: 07111500 Goede.rtf - 2:7 [use case diagrams. You can use..] (39:39) (Super)
Codes: [use case diagrams]
Memos: [ME - 01/12/09 [4]]

use case diagrams. You can use UML quite well. But one should not stick to the rigidness of such techniques or the prescriptiveness; one should be able to add your own things, like to be flexible. And nowadays (inaudible - noise) it's too prescribed. We don't have enough space on those diagrams to write down what its purpose originally was, and that is to capture non-technical information.

Code: User acceptance testing {3-0}

P 8: 08072200 and 08072201-Sindiso.rtf - 8:16 [It is one of the processes act..] (74:74) (Super)
Codes: [User acceptance testing]

P 8: 08072200 and 08072201-Sindiso.rtf - 8:19 [So basically it's the UAT proc..] (76:76) (Super)
Codes: [User acceptance testing]

P 9: 08081800 Kabelo.rtf - 9:13 [hey'll have to do user accepta..] (11:11) (Super)
Codes: [User acceptance testing]

Code: user design approach {1-0}~

P 9: 08081800 Kabelo.rtf - 9:15 [But because we follow the user..] (15:15) (Super)
Codes: [user design approach]

Code: user interface {2-0}

P 2: 07111500 Goede.rtf - 2:6 [user interface] (27:27) (Super)
Codes: [user interface]
Memos: [ME - 01/12/09 [3]]

user interface

P 3: 07111501 Magda.rtf - 3:22 [that user interface sketches th..] (144:144) (Super)
Codes: [user interface]

that user interface sketches that you talk about could be (inaudible - noise) ja. Ja, I do agree with that. I'm sure. If you show them this prototypes or these interface sketches, like you call it, that they would understand it much better. To them the system is the interface,
----

Code: User Interface Problem {1-0}~

P 1: 07110506 Ernest.rtf - 1:20 [But now the major disadvantage..] (55:55) (Super)
Codes: [User Interface Problem]

But now the major disadvantage that I would bring forward is that it sort-of clouds the understanding of the user, especially when it comes to issues like maintenance where you have new requirements that are supposed to be implemented, because the user does not know how the existing methods are actually working, the general problems would be to try and elicit the correct requirements from that individual.

Code: User Involvement {10-2}~

P 1: 07110506 Ernest.rtf - 1:28 [So sitting together there is a..] (67:67) (Super)
Codes: [Language Community] [User Involvement]

So sitting together there is a very important aspect or practice of eliciting the requirements or systems development. You are saying that these three guys should be together. You don't want the systems analyst to meet some guys there, get some documents signed and then go and shift that information into the programme, but you want these three levels to actually sit together. The user must be the centre of all those things.

P 1: 07110506 Ernest.rtf - 1:40 [So the client must be there th..] (107:107) (Super)
Codes: [Quality Measure] [User Involvement]
Memos: [ME - 11/13/08 [1]]

So the client must be there throughout and that is basically for quality check purposes, to ensure that the kind of requirements that the user has are actually met.

P 2: 07111500 Goede.rtf - 2:17 [The other way is to incorporat..] (63:63) (Super)
Codes: [User Involvement]

The other way is to incorporate the clients right through the process and not only in the requirements phase - but to talk to client's right through.

P 2: 07111500 Goede.rtf - 2:34 [Involving your users in all th..] (131:131) (Super)
Codes: [User Involvement]
Memos: [Software application domain]

Involving your users in all the stages, so that we don't spend a lot of time on things that's not important. That's I think - and I look at my students and see what they do wrong. They spend hours and hours programming stuff that the end-user are not interested in - in aesthetics and things. Aesthetics of specific functions or functionality the user is normally not interested in. Their priorities are incorrect.

P 5: 07121401-Barry Myburgh Corrected.rtf - 5:17 [Obviously the most effective w..] (57:57) (Super)
Codes: [Collaborative Approach] [User Involvement]

Obviously the most effective way of trying to ensure that is if we have the client participate in the activities of the project. In other words, a collaborative approach where the client is really a part of the team.

P 7: 08072102 Edwin.rtf - 7:6 [So if you interact with the us..] (49:49) (Super)
Codes: [User Involvement]

DATA SOURCE UNACCESSIBLE!: C:\Documents and Settings\user\My Documents\Working Folder 05 11 2008\PhD Current\VOICE\Transcripted Interviews\08072102 Edwin.rtf

P 9: 08081800 Kabelo.rtf - 9:23 [So the user involvement is alw..] (43:43) (Super)
Codes: [User Involvement]

P 9: 08081800 Kabelo.rtf - 9:29 [If ever, in terms of the walkt..] (53:53) (Super)
Codes: [User Involvement]

P 9: 08081800 Kabelo.rtf - 9:34 [If ever - those requirements d..] (83:83) (Super)
Codes: [quality gate] [User Involvement]

P 9: 08081800 Kabelo.rtf - 9:42 [Because obviously I'll be test..] (103:103) (Super)
Codes: [User Involvement]

Code: user stories {3-3}~

P 1: 07110506 Ernest.rtf - 1:26 [That's an approach that is ver..] (65:65) (Super)
Codes: [user stories]

That's an approach that is very, very powerful that a lot of object-oriented programmers actually use today, especially in the agile world. So with user stories, you'll ask the user to give you a different story of what functionality they would like to be implemented. So it's not a use case, it's not as detailed as a use case but you give them some 6 x 6 inch cards and ask them to write a little functionality that they want implemented. So you're breaking down the requirements into those small user stories. Then you take those user stories now and ask the guy who is going to develop to now try and estimate how long it's going to take to actually develop that. Then you are coming up with some design already as you do that.

P 3: 07111501 Magda.rtf - 3:33 [What they do, they get the req..] (192:192) (Super)
Codes: [user stories]

What they do, they get the requirement on user cards. You know the user stories? I don't ?

P 3: 07111501 Magda.rtf - 3:34 [Normally they're the small car..] (196:196) (Super)
Codes: [user stories]

Normally they're the small cards like - something like this size - and then the users would write their requirements here of the system. And what they do, in the end they collect all the cards and they put a price tag on it. The price in terms of the time it would take and the physical resources they would need and (inaudible - speaking softly) the money. And then they tell the users "what would you like us to do in the following three weeks". And then the users would collect some of those and then it becomes a chunk. So they work only on those requirements and they implement it,
----

Code: Users' perception of system {1-0}

P 8: 08072200 and 08072201-Sindiso.rtf - 8:17 [What we do is whenever we deve..] (74:74) (Super)
Codes: [Users' perception os system]

Code: V-model {2-0}~

P 9: 08081800 Kabelo.rtf - 9:2 [Basically, … in terms of metho..] (7:7) (Super)
Codes: [V-model]

P 9: 08081800 Kabelo.rtf - 9:21 [waterfall or the model which V..] (29:29) (Super)
Codes: [V-model]

Code: walk-throughs {1-0}

P 9: 08081800 Kabelo.rtf - 9:26 [walk-throughs] (51:51) (Super)
Codes: [walk-throughs]

Code: waterfall approach {2-0}

P 5: 07121401-Barry Myburgh Corrected.rtf - 5:1 [For situations where it is pos..] (1:1) (Super)
Codes: [waterfall approach]
Memos: [Suitability of Waterfall approach]

For situations where it is possible to clearly define what is required, and where there is enough time to do effective planning, I would follow a pretty conventional waterfall approach.

P 5: 07121401-Barry Myburgh Corrected.rtf - 5:31 [The one I call "controlled qua..] (113:113) (Super)
Codes: [controlled quality"] [waterfall approach]

The one I call "controlled quality"; the other one I call "crafted quality". Now crafted quality is very much what most people call agile (inaudible - cross-talking) okay; and controlled quality is very much the kind-of conventional waterfall lifecycle approach.

Code: What is the research project addressing? {1-0}

P 9: 08081800 Kabelo.rtf - 9:52 [So, is it more like, in terms ..] (159:159) (Super)
Codes: [What is the research project addressing?]

Code: whiteboards and flyers {2-2}

P 1: 07110506 Ernest.rtf - 1:11 [whiteboards and flyers] (31:31) (Super)

Codes: [whiteboards and flyers]
Memos: [ME - 11/12/08 [2]]

whiteboards and flyers

P 1: 07110506 Ernest.rtf - 1:34 [printable whiteboards] (89:89)   (Super)
Codes: [whiteboards and flyers]

printable whiteboards

Code: workshop {1-0}~

P 7: 08072102 Edwin.rtf - 7:8 [Organise a workshop and then y..] (57:57)   (Super)
Codes: [workshop]

Code: Workshops for communicating system functionality with users {2-1}

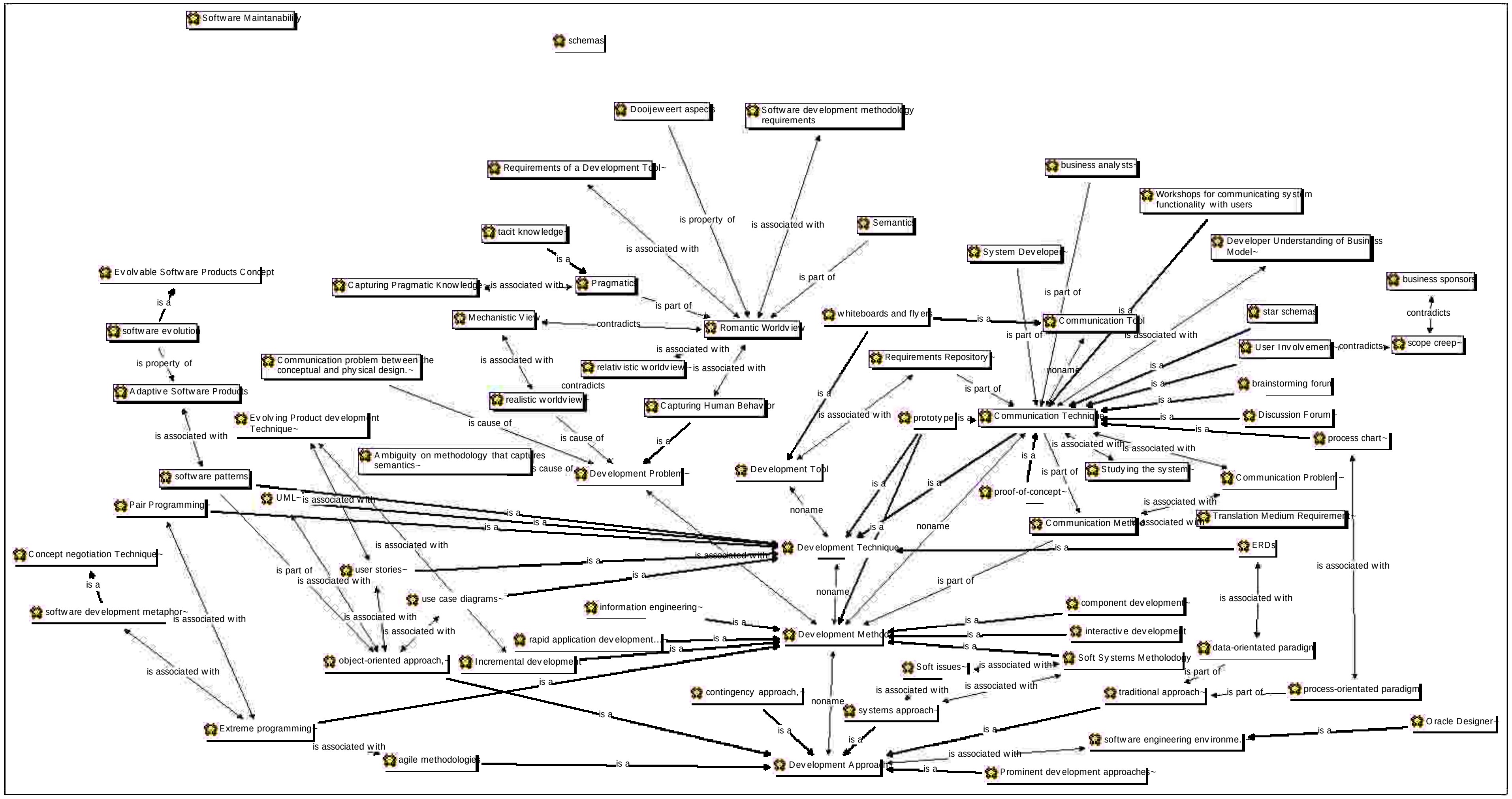
P 1: 07110506 Ernest.rtf - 1:37 [So normally a good number of m..] (101:101)   (Super)
Codes: [Workshops for communicating system functionality with users]

So normally a good number of methodologies that I've studied would have a lot of these workshops where you actually go into the details of how the functionalities of the system are supposed to be. So you find that sometimes you have these workshops and people still remain without the correct understanding.

P 9: 08081800 Kabelo.rtf - 9:24 [Jad sessions, walk-throughs me..] (51:51)   (Super)
Codes: [Workshops for communicating system functionality with users]

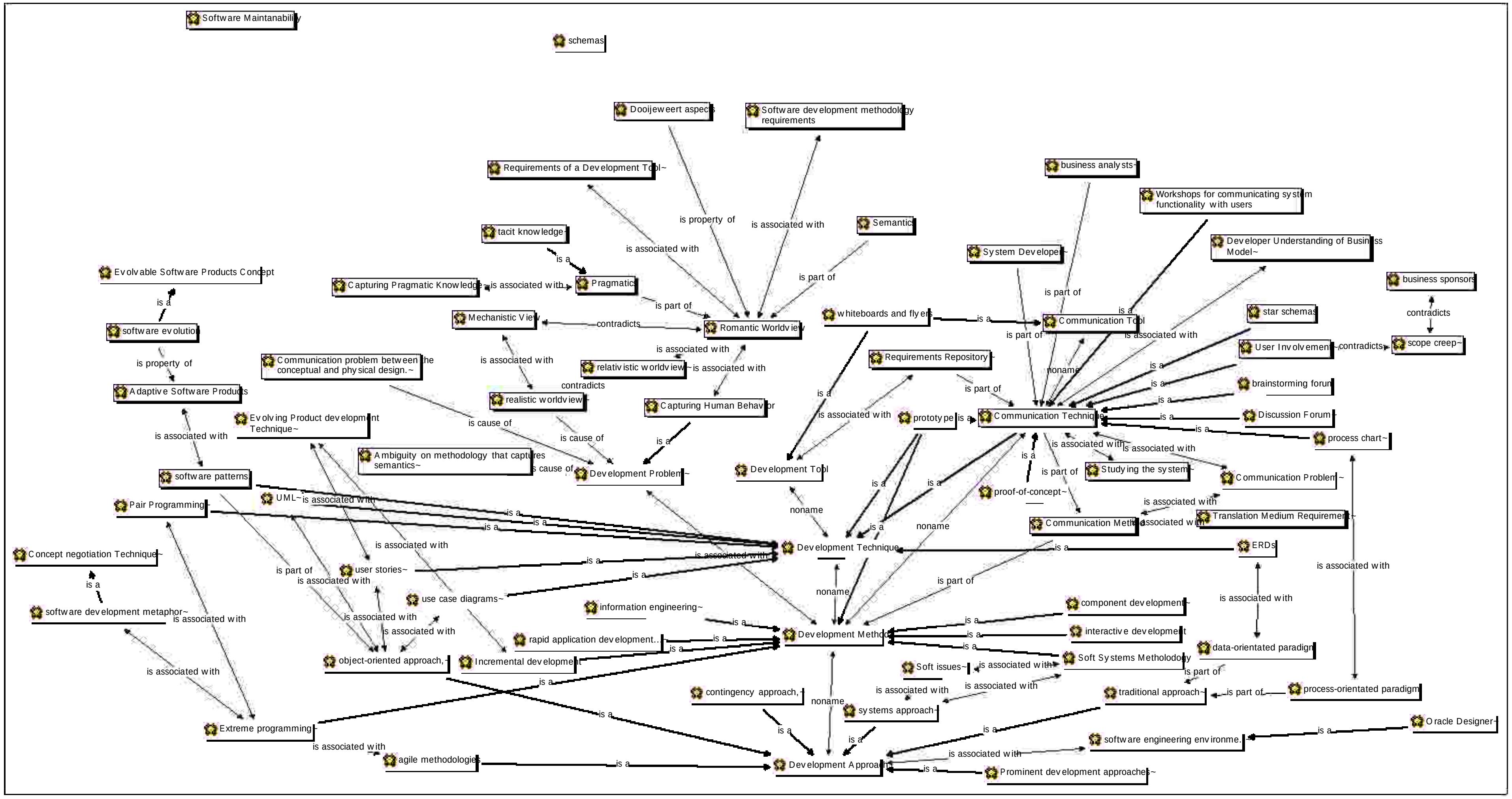
UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Appendix K: Network Diagram-Components of a Software Development Approach



UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Appendix K: Network Diagram-Components of a Software Development Approach



Appendix L: Full List of codes and code families for all the interviews (on CD)

____

HU: An Ontological Approach to Software Development
File: [C:\Documents and Settings\NMavetera\My Docum...\An Ontological Approach to Software Development.hpr5]
Edited by:Super
Date/Time: 10/09/03 05:33:44 PM

____

Code Family: Adaptive Products Development Technique
Created: 08/11/13 02:52:29 PM (Super)
Codes (10): [adaptive evolutionary approach..] [Adaptive Software Products] [Adaptive software products.] [Adaptive Systems] [Adaptive systems development approaches] [configurable systems] [heuristic systems] [Humanist Requirement for Software Development] [intuitive system] [Need to adapt systems to organizational requirements]
Quotation(s): 17

____

Code Family: Communication Method
Created: 08/11/12 02:29:38 PM (Super)
Comment:
This a method needed or used during software development. The method is supposed to improve the way stakeholders share and understand concepts used in the organisation. The purpose of the method is to develop a language community.
Codes (5): [brainstorming forum] [business analysts] [Discussion Forum] [User Involvement] [user stories]
Quotation(s): 19

____

Code Family: Communication Technique
Created: 08/11/12 02:29:16 PM (Super)
Codes (13): [brainstorming forum] [Concept negotiation Technique] [Discussion Forum] [ERDs] [Pair Programming] [proof-of-concept] [prototype] [software development metaphor] [UML] [use case diagrams] [user interface] [user stories] [whiteboards and flyers]
Quotation(s): 27

____

Code Family: Communication Tool
Created: 08/11/12 02:29:52 PM (Super)
Codes (5): [process chart] [UML] [use case diagrams] [user stories] [whiteboards and flyers]
Quotation(s): 12

____

Code Family: Contextual Issues
Created: 08/11/12 02:33:17 PM (Super)
Codes (18): [brainstorming forum] [business analysts] [Capturing Pragmatic Knowledge] [Developer Understanding of Business Model] [Discussion Forum] [Knowledge Sharing] [language database] [Need for a communication medium.] [Ontology Knowledge Base] [Pair Programming] [Process Ontology] [relativistic worldview] [Soft issues] [Soft Systems Metholodogy] [Studying the system] [tacit knowledge] [User Involvement] [user stories]
Quotation(s): 38

____

Code Family: Development Approach
Created: 08/11/12 02:25:59 PM (Super)
Codes (10): [agile methodologies] [contingency approach,] [Incremental development] [New Development Approach] [object-oriented approach,] [Prominent development approaches] [rapid application development...] [Soft issues] [systems approach] [traditional approach]
Quotation(s): 24

____

Code Family: Development Issues

Created: 08/11/12 02:32:36 PM (Super)
Codes (0):
Quotation(s): 0

____

Code Family: Development Method
Created: 08/11/12 02:26:13 PM (Super)
Codes (10): [component development] [data-orientated paradigm] [Extreme programming] [Incremental development] [information engineering] [interactive development] [Pair Programming] [process-orientated paradigm,] [rapid application development...] [Soft Systems Metholodogy]
Quotation(s): 14

____

Code Family: Development Problem
Created: 08/11/12 02:28:05 PM (Super)
Codes (30): [Ambiguity on methodology that captures semantics] [Capturing Human Behavior] [Capturing Pragmatic Knowledge] [Capturing Semantics Issue] [Communication Problem] [Communication problem between the conceptual and physical design.] [Concept negotiation Technique] [Developer Understanding of Business Model] [Development Problem] [Evolving Product development Technique] [Humanist Requirement for Software Development] [Knowledge reuse] [Knowledge Sharing] [Language Community] [language database] [Language Limitations] [Language Requirement] [Need for a communication medium.] [Porblems with case tools] [realistic worldview] [relativistic worldview] [Requirements Repository] [scope creep] [Soft issues] [software evolution] [tacit knowledge] [traditional approach] [Translation Medium Requirement] [User Interface Problem] [User Involvement]
Quotation(s): 75

____

Code Family: Development Technique
Created: 08/11/12 02:26:34 PM (Super)
Codes (17): [brainstorming forum] [Concept negotiation Technique] [data-orientated paradigm] [Discussion Forum] [ERDs] [Evolving Product development Technique] [Extreme programming] [Language Community] [Oracle Designer] [Pair Programming] [software development metaphor] [software patterns.] [UML] [use case diagrams] [User Involvement] [user stories] [Workshops for communicating system functionality with users]
Quotation(s): 43

____

Code Family: Development Tool
Created: 08/11/12 02:27:05 PM (Super)
Codes (4): [language database] [Ontology Knowledge Base] [UML] [whiteboards and flyers]
Quotation(s): 8

____

Code Family: Discussion Tools
Created: 08/11/12 03:17:03 PM (Super)
Codes (1): [brainstorming forum]
Quotation(s): 1

____

Code Family: Evolvable Products Developmet Technique
Created: 08/11/13 02:51:37 PM (Super)
Codes (5): [component development] [Evolvable Software Products Concept] [Evolving Product development Technique] [Incremental development] [software evolution]
Quotation(s): 13

____

Code Family: Interface Issues
Created: 08/11/12 02:32:20 PM (Super)
Codes (3): [User Interface Problem] [User Involvement] [user stories]
Quotation(s): 14

____

Code Family: Knowledge Reuse

Created: 08/11/13 05:01:46 PM (Super)
Codes (2): [Knowledge reuse] [Requirements Repository]
Quotation(s): 3

Code Family: Ontology Requirement
Created: 08/11/13 05:28:55 PM (Super)
Codes (0):
Quotation(s): 0

Code Family: Pragmatics
Created: 08/11/12 02:31:30 PM (Super)
Codes (1): [tacit knowledge]
Quotation(s): 1

Code Family: Semantics
Created: 08/11/12 02:31:39 PM (Super)
Codes (6): [Ambiguity on methodology that captures semantics] [Capturing Semantics Issue] [Concept negotiation Technique] [Humanist Requirement for Software Development] [Ontology Knowledge Base] [Process Ontology]
Quotation(s): 11

Code Family: Syntactics
Created: 08/11/12 02:31:49 PM (Super)
Codes (0):
Quotation(s): 0

Code Family: User Involvement
Created: 08/11/12 02:30:19 PM (Super)
Codes (4): [brainstorming forum] [Discussion Forum] [User Involvement] [user stories]
Quotation(s): 16

Appendix M: Full list of incidents and memos for all the interviews (on CD)

HU: An Ontological Approach to Software Development
File: [C:\Documents and Settings\NMavetera\My Docum...\An Ontological Approach to Software Development.hpr5]
Edited by: Super
Date/Time: 10/09/03 05:50:13 PM

MEMO: Adaptive systems (0 Quotations) (Super, 09/04/23 06:25:14 PM)
No codes
No memos
Type: Memo

Adaptive systems consider the context of the task and change accordingly to suit the environment.

MEMO: Agile Approach (0 Quotations) (Super, 09/04/23 11:09:42 AM)
No codes
No memos
Type: Memo

Agile approach is a methodology suitable for developing systems fast and where time is a very limited resource.

MEMO: Business analyst (0 Quotations) (Super, 09/04/28 07:41:28 PM)
No codes
No memos
Type: Memo

This combination of duties may be used to maintain the business model of the system throughout the developmental stages.

MEMO: Case Tools (0 Quotations) (Super, 09/04/23 06:16:09 PM)
No codes
No memos
Type: Memo

Case tools can both be useful or a hindrance during software development. Time factor, the prescriptiveness of the tools do not help much when it comes to developing systems on time. However, the case tools as an example of a software engineering environment can increase communication during software development and as well as the quality of the end product.

MEMO: Case tools problem (1 Quotation) (Super, 09/01/12 08:24:20 PM)
P 3: 07111501 Magda.rtf:
150-150

No codes
No memos
Type: Memo

Case tools take more time during analysis, the fact being that they do not reuse previous requirements. An ontology driven tool may reuses this knowledge.

P 3: 07111501 Magda.rtf - 3:25 [So it could help. In my experi..] (150:150) (Super)
Codes: [Problems with case tools]
Memos: [Case tools problem]

So it could help. In my experience through my research, I discovered that sometimes the analysis takes longer than what the people or the developers were used to and then they get discouraged. They should just keep with the process, although it takes longer. In the end you save a lot of time because this is generated automatically (inaudible - noise). My students use this for the last three years - they used Oracle Designer. I saw that with them also, they felt "e.g. we are busy - we are just busy with the analysis and design, we can't make progress". Then all of a sudden you'll generate the code and then they see.

MEMO: Communication Problem (0 Quotations) (Super, 09/04/23 11:32:57 AM)
No codes
No memos
Type: Memo

Computers cannot understand language as used by humans. This is because of the tacit nature of most of meanings associated with words.

MEMO: Communication requirements (0 Quotations) (Super, 09/04/28 07:48:29 PM)
No codes
No memos
Type: Memo

A methodology must encompass communication methods that facilitate the development of a shared understanding in the development process. This supports the need of a knowledge base, such as an ontology knowledge base.

MEMO: Communication with Stakeholders (0 Quotations) (Super, 09/04/23 12:57:30 PM)
No codes
No memos
Type: Memo

During the formatory stages of software development, visual tools such as mind maps, projectors can be used during JAD sessions as communication methods. These may become difficult to use when the software has been developed and is ready to be packaged.

MEMO: Document of inputs (0 Quotations) (Super, 09/04/28 02:04:49 PM)
No codes
No memos
Type: Memo

The ontology knowledge base can be used instead of a document of inputs. The knowledge base is crafted from the analysis and contains the analysis, design, implementation models.

MEMO: Expropriation of Meaning (0 Quotations) (Super, 09/04/23 11:15:55 AM)
No codes
No memos
Type: Memo

Many respondents did not understand the concepts used by the researcher. Instead, they would ask the researcher to explain the context with in which the concepts were being used in the research.

MEMO: Formal Part (0 Quotations) (Super, 09/04/23 11:47:39 AM)
No codes
No memos
Type: Memo

System developers more often concentrate on the formal part of the system when implementing the system. They do not consider the informal part, a part that captures the tacit knowledge inherent in organizations as an important aspect during development.

MEMO: Knowledge Base (0 Quotations) (Super, 09/04/23 06:38:35 PM)
No codes
No memos
Type: Memo

There is very little emphasis on the reusability of knowledge in practice. This is supported by the fact that knowledge bases for previously acquired knowledge are nonexistent in many software development organizations. In the contrary however, since knowledge reuse improves the quality and speed at which software can be developed, practitioners should be able to create knowledge bases of their past experiences.

MEMO: Lack of trust (1 Quotation) (Super, 09/01/12 07:54:19 PM)
P 3: 07111501 Magda.rtf:
72-72
No codes
No memos
Type: Memo

There is a lot of mistrust between developers and users. This lack of trust leads to poor communication, resulting in poor requirements gathering.

P 3: 07111501 Magda.rtf - 3:16 [I don’t know about you Nehemiah..] (72:72) (Super)
Codes: [Development Problem]
Memos: [Lack of trust]

I don’t know about you Nehemiah but - there’s sort-of a hostility between users and system developers. The users, they don’t trust the developers and the developers talk down on the user. Okay, and that’s something that’s still alive and well and that bothers me. We are partners' —

MEMO: Language barrier (1 Quotation) (Super, 09/01/12 08:01:20 PM)
P 3: 07111501 Magda.rtf:
84-84
No codes
No memos
Type: Memo

Plain language, understood by all the stakeholders should be used during communication when doing requirements gathering. It is also important to include a business analyst, a person with business orientation to do the analysis. This is like capturing the domain and business model of the system. Then the systems analysts can be incorporated to capture the system requirements.

P 3: 07111501 Magda.rtf - 3:19 [Plain common language; not tec..] (84:84) (Super)
No codes
Memos: [Language barrier]

Plain common language; not technical jargon. Sometimes we don't even - we are not familiar with our own language, the terms we use, the slang that we use. So we should really, really use just plain common language at the level that the user can understand. I think sometimes - you know, nowadays people are very interested in business analysts or business analytics or whatever they call it, where you have business people talking to your users. But I would go further, I wouldn't just have a business person there, I would like a technical person with business knowledge to do this, to do the communication. Ja. Do you have such - I think you have a B.com degree at Mafikeng?

MEMO: Managing scope creep (1 Quotation) (Super, 09/01/12 08:47:38 PM)
P 3: 07111501 Magda.rtf:
214-216
No codes
No memos
Type: Memo

Since user requirements gathering take 80% of the development time, it is important to get a development tool that speeds up the process. Without that, developers risk rushing the requirements gathering process and implementing a wrong system.

P 3: 07111501 Magda.rtf - 3:35 [Is spent on 20 percent of the ..] (214:216) (Super)
Codes: [scope creep]
Memos: [Managing scope creep]

Is spent on 20 percent of the requirement. Are they willing to do that? Are they willing to spend 80 percent of the time? If they cut that 20 percent of the requirement, they could save 80 percent of the time you can go onto a next project. So just discuss it with them. Ask them what their philosophy is, I would say, because some people really want that product and they want it in the way they visualise it and they want to add stuff and so on. And if they are willing to pay for that, fine with me, then I will add it because in the end of the day, it's [them] that are paying for the product, not me.

But other companies are very, very, very driven and they want you to produce quick results and then I would show them. Think about it, if you are in your class also, the amount of time you spend with students in your office. It could be one student that takes up all your time in the office. It's the same with this principle. There's a small amount of requirement and it takes 80 percent of your time.

MEMO: ME - 01/12/09 (1 Quotation) (Super, 09/01/12 02:53:21 PM)
P 2: 07111500 Goede.rtf:
19-19
No codes
No memos
Type: Memo

The problem with traditional approach emphasizes conformance of the system to requirements but negates the needs of the user.

P 2: 07111500 Goede.rtf - 2:2 [I'm very much against the trad..] (19:19) (Super)
Codes: [traditional approach]
Memos: [ME - 01/12/09]

I'm very much against the traditional paradigm because I believe it leads to a case where you have a system that conforms to the requirements and not to the user's needs.

MEMO: ME - 01/12/09 [1] (1 Quotation) (Super, 09/01/12 02:57:13 PM)
P 2: 07111500 Goede.rtf:
19-19
No codes
No memos
Type: Memo

This approach is suitable for developing technical applications.

P 2: 07111500 Goede.rtf - 2:3 [Object-oriented programming to..] (19:19) (Super)
Codes: [object-oriented approach,]
Memos: [ME - 01/12/09 [1]]

Object-oriented programming to me works well in the technical applications. Perhaps it's not yet proven enough for me in today's software development, except perhaps for requirement analysis.

MEMO: ME - 01/12/09 [2] (1 Quotation) (Super, 09/01/12 03:02:43 PM)
P 2: 07111500 Goede.rtf:
23-23
No codes
No memos
Type: Memo

Object -oriented approach techniques are easily understood by the users. They can therefore be used and verified with the users.

P 2: 07111500 Goede.rtf - 2:5 [use case diagrams] (23:23) (Super)
Codes: [use case diagrams]

Memos: [ME - 01/12/09 [2]]

use case diagrams

MEMO: ME - 01/12/09 [3] (1 Quotation) (Super, 09/01/12 03:06:44 PM)
P 2: 07111500 Goede.rtf:
27-27
No codes
No memos
Type: Memo

User interface can be a very good communication tool if well designed. This also can be used to improve the way people use the computer.

P 2: 07111500 Goede.rtf - 2:6 [user interface] (27:27) (Super)
Codes: [user interface]
Memos: [ME - 01/12/09 [3]]

User interface

MEMO: ME - 01/12/09 [4] (1 Quotation) (Super, 09/01/12 03:15:46 PM)
P 2: 07111500 Goede.rtf:
39-39
No codes
No memos
Type: Memo

They can be used to capture the informal part of an information system. However, there is not enough room for developers to add all their information on these diagrams.

P 2: 07111500 Goede.rtf - 2:7 [use case diagrams. You can use..] (39:39) (Super)
Codes: [use case diagrams]
Memos: [ME - 01/12/09 [4]]

Use case diagrams. You can use UML quite well. But one should not stick to the rigidness of such techniques or the prescriptiveness; one should be able to add your own things, like to be flexible. And nowadays (inaudible - noise) it's too prescribed. We don't have enough space on those diagrams to write down what its purpose originally was, and that is to capture non-technical information.

MEMO: ME - 01/12/09 [5] (1 Quotation) (Super, 09/01/12 04:02:17 PM)
P 2: 07111500 Goede.rtf:
43-45
No codes
No memos
Type: Memo

Ontologies fall into the relativistic world view and can be used to bridge the communication gap between the stakeholders in software development.

P 2: 07111500 Goede.rtf - 2:10 [Ja. Well ontologies - I'm a bi..] (43:45) (Super)
No codes
Memos: [ME - 01/12/09 [5]]

Ja. Well ontologies - I'm a bit of a philosopher as well - and it's actually an easy answer but very difficult to implement but it's between the realistic and the relativistic worldview. The realistic worldview says that everything is agreeable. You and I can decide this is a mouse and this is the function of the mouse. That is the realistic worldview and that is where the problem with the engineers comes in. They follow a realistic worldview. They think they agree on the purpose of this thing.

And if we follow a relativistic worldview, you might have an opinion on this device and I can have a different opinion on this device. And if we then get down to a discussion, we might design a much better mouse than this - if we sit down and think "what is the real purpose".

MEMO: ME - 01/12/09 [6] (1 Quotation) (Super, 09/01/12 05:06:16 PM)
P 2: 07111500 Goede.rtf:
103-103
No codes
No memos
Type: Memo

Scope creep can be reduced by getting participation or involvement of the right people, including the business sponsor. The right people involvement ensures also that the appropriate and correct requirements are gathered.

P 2: 07111500 Goede.rtf - 2:30 [I think scope creep always hap..] (103:103) (Super)
No codes
Memos: [ME - 01/12/09 [6]]

I think scope creep always happens but the extent of it depends on your - two things - actually mainly on the quality of your requirements. If your requirements are done well and you involve the right people, you might have a situation where you don't get that much scope through it. And then as - if you are using a methodology where your end-user is involved, then scope should be their problem, not your problem in a sense. But that's only the case if you have a end-user or a - I don't want to use the word end-user, I want to use a word (indiscernible - audio quality) user in data warehousing, business sponsors - somebody higher up in the organisation. Somebody more important than the secretary that's going to type on the system; somebody that has power inside the organisation. I want to have that person involved and he should prioritise all these additional scope requirements, not the programmers.

MEMO: ME - 04/23/09 (0 Quotations) (Super, 09/04/23 11:41:12 AM)
No codes
No memos
Type: Memo

MEMO: ME - 04/28/09 (0 Quotations) (Super, 09/04/28 01:06:12 PM)
No codes
No memos
Type: Memo

The documentation should be captured and stored in a repository. This will be a knowledge base upon which future problems can be refereed for solutions. Think of an ontology knowledge base which also captures the human aspects of organizational information systems.

MEMO: ME - 04/28/09 [1] (0 Quotations) (Super, 09/04/28 02:09:11 PM)
No codes
No memos
Type: Memo

The different types of testing do not however ensure schedule cost issues are adhered to. They can of course ensure syntactic quality of the system.

MEMO: ME - 04/28/09 [3] (0 Quotations) (Super, 09/04/28 07:50:19 PM)
No codes
No memos
Type: Memo

MEMO: ME - 11/12/08 (1 Quotation) (Super, 08/11/12 02:21:41 PM)
P 1: 07110506 Ernest.rtf:
25-25
No codes
No memos
Type: Memo

Agile Approaches are a development methodology that uses extreme programming as a method and pair programming as a technique for XP. This PP allows a shared understanding of the development area.

P 1: 07110506 Ernest.rtf - 1:8 [For example, we are developing..] (25:25) (Super)
No codes
Memos: [ME - 11/12/08]

For example, we are developing using agile methodologies which is my area of interest. In Extreme programming you have a practice that is called Pair Programming. In Pair Programming you put two people - two programmers - on one machine and that way, that knowledge which cannot be written down but which is in those developers' minds, can actually be exchanged as they do the development.

MEMO: ME - 11/12/08 [1] (1 Quotation) (Super, 08/11/12 03:06:20 PM)
P 1: 07110506 Ernest.rtf:
33-33
No codes
No memos
Type: Memo

The software development metaphor allows people to negotiate meaning of concepts amongst themselves to achieve a shared understanding. This is a requirement to ensure that all the stakeholders in the software development process build a language community

P 1: 07110506 Ernest.rtf - 1:14 [So instead of people being stu..] (33:33) (Super)
Codes: [Concept negotiation Technique] [Language Community]
Memos: [ME - 11/12/08 [1]]

So instead of people being stuck to those technical terms and so forth, they come up with some naming convention that is common to everyone. And that naming convention brings everyone to the same language, despite the differences and so forth.

MEMO: ME - 11/12/08 [2] (1 Quotation) (Super, 08/11/12 03:12:46 PM)
P 1: 07110506 Ernest.rtf:
31-31
No codes
No memos
Type: Memo

The software development process requires a discussion forum or a knowledge sharing platform where people can exchange their knowledge without being pestered.

P 1: 07110506 Ernest.rtf - 1:11 [whiteboards and flyers] (31:31) (Super)
Codes: [whiteboards and flyers]
Memos: [ME - 11/12/08 [2]]

Whiteboards and flyers

MEMO: ME - 11/12/08 [3] (1 Quotation) (Super, 08/11/12 03:24:54 PM)
P 1: 07110506 Ernest.rtf:
43-43
No codes
No memos
Type: Memo

To capture tacit knowledge, people require to have methods that capture the soft issues of organizational information systems. These employ the Soft Systems Methodologies that considers the humanist issues inherent in organizations. Domain ontologies can feel this gap. One can develop a domain ontology repository and use it in IS in place of the schema.

P 1: 07110506 Ernest.rtf - 1:17 [So I think the best way to act..] (43:43) (Super)
Codes: [New Development Approach]
Memos: [ME - 11/12/08 [3]]

So I think the best way to actually represent that information will be to come up with some - not a framework - but maybe language of some sort that has the ability to actually express those concepts and to relate to technical issues that way. So I'm not very sure of what language you could actually use for that but I'm tending to think in terms of the approaches to development.

MEMO: ME - 11/12/08 [4] (0 Quotations) (Super, 08/11/12 03:31:05 PM)
No codes
No memos
Type: Memo

MEMO: ME - 11/12/08 [5] (1 Quotation) (Super, 08/11/12 03:31:14 PM)
P 1: 07110506 Ernest.rtf:
47-47
No codes
No memos
Type: Memo

As much as it is an improved technique to software development, it falls short of capturing semantics. This technique which falls into the object oriented traditional paradigm however moves from functions to entities in organizational systems. It encourages reusability of objects though.

P 1: 07110506 Ernest.rtf - 1:18 [So it's more of the - when you..] (47:47) (Super)
Codes: [UML]
Memos: [ME - 11/12/08 [5]]

So it's more of the - when you look at an object, it gives a better understanding of the system or whatever you're dealing with but I wouldn't call it semantics.

MEMO: ME - 11/12/08 [6] (1 Quotation) (Super, 08/11/12 06:56:48 PM)
P 1: 07110506 Ernest.rtf:
61-61
No codes
No memos
Type: Memo

The use of a developer instead of analyst, programmer in development reduces the gap in understanding user requirements. The developer does both the analysis and programming of the system.

P 1: 07110506 Ernest.rtf - 1:25 [Ja, the approach that you are ..] (61:61) (Super)
No codes
Memos: [ME - 11/12/08 [6]]

Ja, the approach that you are defining, I generally call it a traditional approach, where you have a systems analyst, you have a programmer, and then you have got the business guy, the user. I call that the traditional approach. In the approach that I've studied in detail and also used, you'll find that the gap between the analyst and the programmer is actually closed by coming up with somebody who's called a developer, who actually elicits the requirements directly from the user. And that user, in the methods that I'm actually talking about, becomes part of the development team which means that those details - those metaphors that I was talking about - the language that is used by the team becomes familiar to that user. And the problem - the architectural problems where the systems analyst has some design diagrams and whatever that may not be clear to the programmer —

MEMO: ME - 11/12/08 [7] (1 Quotation) (Super, 08/11/12 07:09:35 PM)

P 1: 07110506 Ernest.rtf:
67-67
No codes
No memos
Type: Memo

User stories can be used for time budgeting purposes because each story can be implemented on its own. This also improves the communication between analyst, developer and users in a way established better understanding of the system.

P 1: 07110506 Ernest.rtf - 1:29 [So it is those kinds of tools ..] (67:67) (Super)
No codes
Memos: [ME - 11/12/08 [7]]

So it is those kinds of tools where the user, the analyst - whoever we call the analyst, and the programmer sits together. So sitting together there is a very important aspect or practice of eliciting the requirements or systems development. You are saying that these three guys should be together. You don't want the systems analyst to meet some guys there, get some documents signed and then go and shift that information into the programme, but you want these three levels to actually sit together. The user must be the centre of all those things. You're saying the user has to tell you the kind of requirements that you are talking about, help them mainly in terms of how to represent that information but you want them to give you that, and also to prioritise that as you develop this system, "we want you to start with the following things". And then you take those now and give them to the developer now and say "okay, let's try and find out how long it's going to take. So can you give estimates of how long each of those user cards is going to take to develop"?

MEMO: ME - 11/12/08 [8] (1 Quotation) (Super, 08/11/12 07:15:01 PM)
P 1: 07110506 Ernest.rtf:
73-73
No codes
No memos
Type: Memo

Communication medium, that is understood by both developers and users should be found and used. This really improves the understanding and the knowledge sharing in software development.

P 1: 07110506 Ernest.rtf - 1:30 [One group of clients were clie..] (73:73) (Super)
Codes: [Need for a communication medium.]
Memos: [ME - 11/12/08 [8]]

One group of clients were clients who really don't know much about systems development.

MEMO: ME - 11/12/08 [9] (1 Quotation) (Super, 08/11/12 07:30:36 PM)
P 1: 07110506 Ernest.rtf:
91-91
No codes
No memos
Type: Memo

Ontology has an open architecture and can actually be adapted to different scenarios. The ability to assign context depended meaning and hence functionality to an ontology driven case tool is a big plus.

P 1: 07110506 Ernest.rtf - 1:35 [The other thing that I would a..] (91:91) (Super)
Codes: [Requirements of a Development Tool]
Memos: [ME - 11/12/08 [9]]

The other thing that I would also mention about the competence of the tool, is that the tool should not be too much restricted to a particular methodology because, like you said, a methodology can actually change, people can come up with different approaches to that methodology, and change the framework and its parameters. So when that happens, the tool itself must actually be adaptable to that.

MEMO: ME - 11/13/08 (1 Quotation) (Super, 08/11/13 11:21:41 AM)
P 1: 07110506 Ernest.rtf:
97-97
No codes
No memos
Type: Memo

The software development approach must not be restricted to a single development tool. It will limit the acceptance of the tool itself by the users as well as negating the importance skill set variability in developers.

P 1: 07110506 Ernest.rtf - 1:36 [that you are dealing with peop..] (97:97) (Super)
Codes: [Need a Variety of Development Tools]
Memos: [ME - 11/13/08]

that you are dealing with people and people are important in those systems, and what they know is also important. It means that if you try to come up with tools like those, you are threatening the development approach itself. The methodology is threatened because you are restricting to it a specific tool within the project. Remember the projects are different and the skill sets of people that you work with are different.

MEMO: ME - 11/13/08 [1] (1 Quotation) (Super, 08/11/13 01:32:40 PM)
P 1: 07110506 Ernest.rtf:
107-107
No codes
No memos
Type: Memo

User involvement throughout the project development process in agile methodologies also ensures quality of the product. The user is able to check the conformance of the requirements and sign off the different stages on the run.

P 1: 07110506 Ernest.rtf - 1:40 [So the client must be there th..] (107:107) (Super)
Codes: [Quality Measure] [User Involvement]
Memos: [ME - 11/13/08 [1]]

So the client must be there throughout and that is basically for quality check purposes, to ensure that the kind of requirements that the user has are actually met.

MEMO: ME - 11/13/08 [3] (1 Quotation) (Super, 08/11/13 04:45:10 PM)
P 1: 07110506 Ernest.rtf:
153-153
No codes

No memos
Type: Memo

The software development fraternity should develop a tool or language that is capable of capturing the human aspect of communication. This will capture the informal part of the system into the software product. {Ontology Aspect Can help}

P 1: 07110506 Ernest.rtf - 1:55 [because the language is the on..] (153:153) (Super)
Codes: [Language Requirement] [New Development Approach]
Memos: [ME - 11/13/08 [3]]

because the language is the only solid thing or it's the only output that you get from a human being that lets you know what is happening in their brain. And if you want to deal with the informal part of the system, you are really looking at the softer issues, the human issues of software development. The behaviour of human beings and their languages would be the centre that people have to deal with in order to understand.

MEMO: ME - 11/13/08 [4] (1 Quotation) (Super, 08/11/13 04:57:01 PM)
P 1: 07110506 Ernest.rtf:
155-155
No codes
No memos
Type: Memo

This supports the idea that language limitations are the most inhibiting factor that limits software products to capture informal requirements in systems. This language should also allow the building of a language community and facilitate knowledge sharing using concepts.

P 1: 07110506 Ernest.rtf - 1:56 [For example, when somebody exp..] (155:155) (Super)
Codes: [Language Limitations]
Memos: [ME - 11/13/08 [4]]

For example, when somebody expresses their idea, the limitations of that expression or that representation, are within the language that they are using.

MEMO: ME - 11/13/08 [5] (1 Quotation) (Super, 08/11/13 04:58:45 PM)
P 1: 07110506 Ernest.rtf:
155-155
No codes
No memos
Type: Memo

Maybe since ontologies are computer processable and can capture the domain knowledge of a system, they can fill the language requirement gap discussed herein

P 1: 07110506 Ernest.rtf - 1:57 [So that is exactly the demise ..] (155:155) (Super)
Codes: [Capturing Semantics Issue] [Language Limitations]
Memos: [ME - 11/13/08 [5]]

So that is exactly the demise that we have as software developers, that if we try to go into the details of the softer side - the non-formal parts of the system - we find that we have limitations in terms of the language. Because whatever language you are using, when I try to use now the computer language to express that, it might not represent what you are talking about. So it is really the issues of language that should be dealt with and so forth.

MEMO: ME - 11/13/08 [6] (1 Quotation) (Super, 08/11/13 05:18:53 PM)
P 1: 07110506 Ernest.rtf:
161-161
No codes
No memos
Type: Memo

Ontology concept can be used to understand meanings of concepts. It's an ontology knowledge base that can be consulted by developers in a domain and compare the different meanings from different organizations.

P 1: 07110506 Ernest.rtf - 1:58 [But what usually happens is th..] (161:161) (Super)
Codes: [Knowledge Sharing] [Language Community]
Memos: [ME - 11/13/08 [6]]

But what usually happens is that when you get to an organisation, you need to understand their language, the way they talk about things, whether they are technical or non-technical but for the purposes of your requirements. The way the information is represented will sometimes differ from other organisations even in the same field.

MEMO: ME - 11/13/08 [7] (0 Quotations) (Super, 08/11/13 05:35:23 PM)
No codes
No memos
Type: Memo

Knowledge

MEMO: ME - 11/13/08 [8] (1 Quotation) (Super, 08/11/13 05:37:24 PM)
P 1: 07110506 Ernest.rtf:
165-165
No codes
No memos
Type: Memo

If working with processes, process ontologies can be used to capture the knowledge in organizational processes and ontologies can come in place of the syntactic schema.

P 1: 07110506 Ernest.rtf - 1:61 [So what I'm saying is that you..] (165:165) (Super)
Codes: [Ontology Knowledge Base] [Process Ontology]
Memos: [ME - 11/13/08 [8]]

So what I'm saying is that you need some formal way where you can have a database that has some schemas of those words, of those concepts that we're talking about. For example, if we talk about a loan, all the financial institutions will have something called a loan. So you can have a loan schema, you find that your Nedbank, your ABSAs and Standard Banks might actually have different terms for your interest - for the interest rate, for whatever, for the borrowing rate or whatever - you know, those technical terms that they use. They might have different terms that actually mean the same thing, so your schema must be able to capture that - ensure that it is the same thing so that when you now check back and re-use it in different institutions you are not really changing much. It will come out there according to their language but you are still using the same thing

---

MEMO: Need for methodology (1 Quotation) (Super, 09/01/12 08:29:20 PM)
P 3: 07111501 Magda.rtf:
160-160
No codes
No memos
Type: Memo

A methodology dictates the way a software engineering environment is subsequently used. Without that, introduces the software engineering environment on its own may cause problems of fit. This must be the case with all these ontology driven software engineering environments. The problem facing developers is a lack of a ontology driven methodology that directs most of the tools developed so far.

P 3: 07111501 Magda.rtf - 3:27 [And another thing I would also..] (160:160) (Super)
No codes
Memos: [Need for methodology]

And another thing I would also be careful though is, if companies buy into one of those software engineering environments - say for example Oracle Designer - that software engineering environment is essentially a companion to a methodology. For example, Oracle Designer, the companion methodology I would say is like information engineering. They call it something else but in essence it's information engineering. And if that methodology is not understood and in place in the company, and you use this software engineering environment, it's difficult. They find it difficult to do that. But if they know the methodology and they buy into this software engineering environment, then it becomes (inaudible - speaking softly). I call a software engineering environment and a methodology companion, those should go together. You can't have the one without the other one.

MEMO: Popular Methodologies (0 Quotations) (Super, 09/04/28 06:40:11 PM)
No codes
No memos
Type: Memo

The waterfall and V-Model are popular methodologies in the banking sector because of their repetitive and structured nature

---

MEMO: Problems in Software development (0 Quotations) (Super, 09/04/23 01:09:16 PM)
No codes
No memos
Type: Memo

The software development process faces a problem of user involvement due to time and business requirements of the users. More often, they do not have time to sit in the project teams and as a result, they do not understand fully what the system needs to do.

MEMO: Problems of not Involving software tester (0 Quotations) (Super, 09/04/28 08:09:51 PM)

No codes
No memos
Type: Memo

The software testers as people who check the fidelity of the implemented system to the business requirements should be involved during requirements specification. This helps them to check the feasibility of the test cases against the user requirements.

MEMO: Prominent development approaches (1 Quotation) (Super, 09/01/12 07:35:51 PM)
P 3: 07111501 Magda.rtf:
43-43
No codes
No memos
Type: Memo

Structured and object oriented approaches are widely used in industry.

P 3: 07111501 Magda.rtf - 3:9 [You mentioned only two of thos..] (43:43) (Super)
Codes: [Prominent development approaches]
Memos: [Prominent development approaches]

You mentioned only two of those approaches and in my research I saw that those two are the mostly used approaches in the industry. You find that there's a movement towards rapid application development but the structured and the object-orientated approach are indeed, I think, the most used approaches.

MEMO: Requirements problem (1 Quotation) (Super, 09/01/12 07:49:53 PM)
P 3: 07111501 Magda.rtf:
66-66
No codes
No memos
Type: Memo

Requirements are not fully captured because of communication problems as well as time limitations.

P 3: 07111501 Magda.rtf - 3:13 [And I - really I would argue t..] (66:66) (Super)
Codes: [Development Problem]
Memos: [Requirements problem]

And I - really I would argue that it's because the (client can't??) define the requirements correctly because we don't understand what exactly what this user wants. So if we put more effort into requirements gathering, I think you will see that problem - I wouldn't say it'll be solved - but much better. I think we are so much in haste to get the product delivered, that we don't even bother to look at the requirements.

MEMO: Requirements Repository (1 Quotation) (Super, 09/01/12 04:47:03 PM)
P 2: 07111500 Goede.rtf:
75-75
No codes
No memos
Type: Memo

A development tool must have a requirements repository that captures and stores user requirements during analysis, as an analysis model. These repository must be able to be consulted at every stage of the system development. Besides improving on requirements communication throughout the project, this increases also the time to market, quality of the software products.

P 2: 07111500 Goede.rtf - 2:24 [Firstly because I think it's m..] (75:75) (Super)
Codes: [Need for a development tool] [Requirements Repository]
Memos: [Requirements Repository]

Firstly because I think it's mostly - let me say, I don't think the end-user will benefit as much as the developer from doing it because you have one version of the truth of the requirement, you have one place - central repository - where you can find the requirements, you can set up checklists to see if you have fulfilled the requirements, the requirements are (nearby??) but then the system must force the developers or the analysts to fill it in.

MEMO: Requirements reusability (0 Quotations) (Super, 09/04/28 12:56:52 PM)
No codes
No memos
Type: Memo

It is difficult to reuse requirements across organizations. However, a framework of applications that run in the organizations can be development and later adapted to different organizational requirements. Such frameworks can be software kernels or SPL.

MEMO: Service oriented architecture (1 Quotation) (Super, 09/01/12 05:01:54 PM)
P 2: 07111500 Goede.rtf:
99-99
No codes
No memos
Type: Memo

This a concept or technique that encourages reuse of business models. The risk which developers are faced with is that, they tend to force business models where they can not fit.

P 2: 07111500 Goede.rtf - 2:28 [service oriented architecture] (99:99) (Super)
No codes
Memos: [Service oriented architecture]

service oriented architecture

MEMO: Software application domain (1 Quotation) (Super, 09/01/12 05:31:19 PM)
P 2: 07111500 Goede.rtf:
131-131
No codes
No memos
Type: Memo

User involvement helps to capture the context and application domain of a software product. This can be likened to a knowledge base that is populated with domain ontologies and task ontologies.

P 2: 07111500 Goede.rtf - 2:34 [Involving your users in all th..] (131:131) (Super)
Codes: [User Involvement]
Memos: [Software application domain]

Involving your users in all the stages, so that we don't spend a lot of time on things that's not important. That's I think - and I look at my students and see what they do wrong. They spend hours and hours programming stuff that the end-user are not interested in - in aesthetics and things. Aesthetics of specific functions or functionality the user is normally not interested in. Their priorities are incorrect.

MEMO: Software Crises (0 Quotations) (Super, 09/04/23 06:52:52 PM)
No codes
No memos
Type: Memo

For developers to have a product that is functional on time, they do not emphasize on the cost reduction, timely delivery, and quality requirements of system development.

MEMO: Software development methodology requirements (0 Quotations) (Super, 09/01/12 05:16:20 PM)
No codes
No memos
Type: Memo

Goede, argues that, for a development methodology to capture semantics, it must have the following characteristics: be able to capture the human factor inherent in organizational systems, direct developers to ask the right questions, must be grounded in the relativistic worldview, must incorporate tool that use the soft systems methodologies, and the techniques and tools so used must facilitate discussion and not assume understanding. Also, the methodology must allow a stratified, hierarchical way of describing a scenario. The must be several levels or ways of describing a scenario, as proposed by Dooijeweert. The various system ontologies, domain, status, process etc could be the answer to this.

MEMO: Software Migration (0 Quotations) (Super, 09/04/28 11:49:29 AM)
No codes
No memos
Type: Memo

The amount of work and tasks that are needed to migrate from one system to the other (new) discourage much organization from changing their systems. In a way, the systems are not adapted to the changing environment. This calls upon developers to look for a method that enables smooth development of adaptive software systems.

MEMO: Structured development approach (1 Quotation) (Super, 09/01/12 07:22:59 PM)
P 3: 07111501 Magda.rtf:
25-25
No codes
No memos
Type: Memo

Process Oriented and data oriented software development are referred to as paradigms. However, I would put them under the structured development approach.

P 3: 07111501 Magda.rtf - 3:3 [In the process-orientated para..] (25:25) (Super)
Codes: [traditional approach]
Memos: [Structured development approach]

In the process-orientated paradigm, the philosophical view of the developers of these methodologies is that the process is the most important aspect of the system. And the data-orientated paradigm is where they say [that] data stays constant, so we should rather focus on the data instead of the process, it would make our lives much easier.

MEMO: Studying the business environment (1 Quotation) (Super, 08/11/13 02:21:11 PM)
P 1: 07110506 Ernest.rtf:
103-103
No codes
No memos
Type: Memo

These workshops are synonymous with JAD sessions. As they are helping with knowledge sharing, they also improve the way users and developers understand the system. The important tenet is to enable analysts to understand and capture the business model of the system.

P 1: 07110506 Ernest.rtf - 1:38 [And besides that, when you do ..] (103:103) (Super)
Codes: [Developer Understanding of Business Model] [Knowledge Sharing]
Memos: [Studying the business environment]

And besides that, when you do such workshops, you are not just aiming at them understanding the system but you're also aiming at the developer's understanding of the business model that the organisation is using.

MEMO: Suitability of Waterfall approach (1 Quotation) (Super, 09/04/02 07:53:20 PM)
P 5: 07121401-Barry Myburgh Corrected.rtf:
1-1
No codes
No memos
Type: Memo

The waterfall approach is best suited for situations where there is more time for effective planning.

P 5: 07121401-Barry Myburgh Corrected.rtf - 5:1 [For situations where it is pos..] (1:1) (Super)
Codes: [waterfall approach]
Memos: [Suitability of Waterfall approach]

For situations where it is possible to clearly define what is required, and where there is enough time to do effective planning, I would follow a pretty conventional waterfall approach.

MEMO: System Developer (1 Quotation) (Super, 09/01/12 04:36:00 PM)

P 2: 07111500 Goede.rtf:
71-71
No codes
No memos
Type: Memo

Software companies are engaging the services of developers to act as communication medium. A developer has other technical and business skills and conveying messages amongst stakeholders will become easier since these people understand both languages.

P 2: 07111500 Goede.rtf - 2:23 [Now I've seen a number of orga..] (71:71) (Super)
Codes: [System Developer]
Memos: [System Developer]

Now I've seen a number of organisations employing people that has a little bit of technical skill a lot of business skills to serve as the interface between the technical people and the users.

MEMO: Time Box (0 Quotations) (Super, 09/04/23 07:15:43 PM)
No codes
No memos
Type: Memo

This is an agile development technique that fixes the time to which a software product can be developed.

MEMO: User acceptance testing (0 Quotations) (Super, 09/04/28 12:22:40 PM)
No codes
No memos
Type: Memo

User acceptance testing can be used in an attempt to make user understand what the system will do. It however comes at the end of the development stage and cannot be used to ensure that the users have an overall understanding of the whole system. In other words, it does not maintain the user understanding of the system throughout the development stages. It is in this case used as a rubber stamp of what the developers think the users wanted.

MEMO: User Involvement (0 Quotations) (Super, 09/04/23 01:03:36 PM)
No codes
No memos
Type: Memo

The client can understand the way a software product is used better if they are involved from the start of the project up to its commissioning.

MEMO: User Participation (0 Quotations) (Super, 09/04/28 07:38:30 PM)
No codes
No memos
Type: Memo

Users can be used to check the business model of the system through the system functionality

---

MEMO: User understanding (1 Quotation) (Super, 09/01/12 04:27:05 PM)
P 2: 07111500 Goede.rtf:
    67-67
No codes
No memos
Type: Memo

These methodologies ensure that the stakeholders communicate face to face with each other. This enhances the client understanding of the would be system and eventually the quality of the software product.

P 2: 07111500 Goede.rtf - 2:22 [extreme programming and new de..] (67:67)   (Super)
Codes: [agile methodologies]
Memos: [User understanding]

extreme programming and new development,

MEMO: User understanding of the system (1 Quotation) (Super, 09/01/12 04:22:41 PM)
P 2: 07111500 Goede.rtf:
    63-63
No codes
No memos
Type: Memo

These techniques help ensure that the client understanding of the system is maintained throughout the development stages.

P 2: 07111500 Goede.rtf - 2:20 [Ja, I think there are two ways..] (63:63)   (Super)
No codes
Memos: [User understanding of the system]

Ja, I think there are two ways and that is to do a very good verification after the requirements has been gathered. That can be done, once again, with the proof-of-concept. The other way is to incorporate the clients right through the process and not only in the requirements phase - but to talk to client's right through. Incremental development helps a lot to do delivery in smaller pieces, then if you go wrong you don't have to backtrack the whole system. So an iteration and an incremental development helps a lot. But the best way is to have a user onboard, such as all these agile methodologies, they work like this.
