Bridging the Communications Gap between Systems and Software Engineering[1]

Joseph Kasser D.Sc., C.Eng., CM.
Systems Engineering and Evaluation Centre
University of South Australia
School of Electrical and Information Engineering F2-37
Mawson Lakes Campus
South Australia 5095
Telephone: +61 (08) 830 23941, Fax: +61 (08) 830 24723
Email: joseph.kasser@unisa.edu.au

Sharon Shoshany
School of Computer and Information Science
University of South Australia,
Mawson Lakes Campus,
Mawson Lakes, SA, 5095
Australia
Email Sharon.Shoshany@unisa.edu.au

Abstract. Kasser and Shoshany (2000) identified that differences in communications between systems and software engineers were a cause of project failure[2]. This paper introduces a methodology for use by integrated process teams (IPT) to bridge the communications gap between systems and software engineering.

The meta-model for the development process (Kasser and Shoshany 2000) shows that engineers use pattern matching techniques. However, systems and software engineers tend to use different patterns. This paper shows that the sharing of patterns may bridge the communications gap. For it is by sharing of mental patterns that the engineers can understand and communicate the impact of a concept on their discipline. The paper also provides some examples of how patterns may be shared in an integrated design team environment.

BACKGROUND

The current trend towards integrated teams and compliance to process standards means that systems and software engineers are formally required to work together in most of the phases of the system life cycle (SLC). While working together and practicing "active listening" the authors realized **that a communications gulf exists between systems and software** engineers and felt that the gulf

- Tends to be ubiquitous but is not recognized as such.
- Is an unidentified but important contributor to project failures.
- Should be identified and its existence brought to the notice of the systems and software engineering communities.

At this time, the authors also came to the realization that much of the communication gap was due to underlying barriers that included

- The Bodies of Knowledge for the professions
- The role of systems engineer in the SLC.
- Training and Background Differences
- A lack of respect for the other's profession.
- Semantics and the use of language
- Different Concepts.

[1] This work was partially funded from the DSTO SEEC Centre of Expertise Contract.
[2] Failure is defined as cancellation or massive cost and schedule overruns.

A prior paper was written summarizing the analyses of the barriers (Kasser and Shoshany 2000) and outlining the causes of the barriers. This paper follows-on and

- Focuses on bridging the barriers raised by the use of semantics and language as well as the difference in concepts that may remain unknown due to the apparent correctness of words in context.
- Introduces a simple but effective methodology for bridging the gap and maximizing the transfer of meaning between systems and software engineers[3].

THE META-MODEL

While working together, after some heated discussion[4] the authors developed a conceptual meta-model of the system development process within the SLC (Kasser and Shoshany 2000). The meta-model, which became our Rosetta Stone, summarizes the development process in the following manner:

When faced with the problem of meeting the customer's needs, good engineering practice dictates the use of one of two implementation choices

1. **The problem is similar to other problems that have been solved in the past.** Thus this time around, providing a similar solution may solve the problem. The process then becomes one of identifying the applicability of the solutions of the past, to the problem of the present and applying the elements of one or more solutions of the past to solve the problem of the present.
2. **The problem is unique so there are no known solutions.** The process then becomes one of identifying a solution that makes the maximum use of existing solutions to past problems (components) and the minimum use of components to be developed, so as to reduce the risk of failure to deliver on time and within budget.

In addition, engineers don't always reuse solutions or components that worked in the past they reinvent them or try to invent new ones. This practice may result in them turning a 'choice 1' situation into a 'choice 2' situation with corresponding budget and schedule costs while rediscovery takes place.

PATTERN MATCHING IS THE KEY

The key element in bridging the gap is to use active listening enhanced by "pattern matching" during discussions. For example, during a meeting discussing a scenario, design issue or requirement, they can often be seen to be similar to previous encounters.

The approach suggested in this paper is to enhance active listening with pattern matching to improve interpersonal communications. Thus when faced with a problem, the project team should first review (Kasser and Shoshany 2000) and then this paper. These papers will sensitize them to the issues and the potential barriers to communications. Only then should the discussions take place. During these discussions different people may recognize different similarities to previous encounters. Let each take a turn in explaining what pattern they recognize and why. Thus for example

- Systems engineering may recognize a Type A scenario.
- Hardware engineering may recognize a Type B situation.
- Software engineering may see it as a Type C.
- Reliability engineering may see it as a cross between a Type B and a Type C.

Participants in the meeting should use active listening techniques enhanced by pattern matching to apply feed back to the communications process to maximize the probability of sharing the meaning as described below.

ACTIVE LISTENING

Active listening is a standard technique for applying the feedback principle to inter-personal communications to minimize errors in conveying the meaning from one person to another. Active listening first recognizes that during a conversation, most people do not listen to what the other person is saying. They are too busy planning what they will say when the other person pauses. Standard active listening comprises the following multi-step process

1. When the other person speaks give your them full attention and look them straight in the eyes.
2. Begin iteration loop.
3. Listen to everything the other person says and try to understand it fully.
4. Ask questions to clarify anything you don't understand and analyze the response.
5. Rephrase what you have heard in your own words and ask the speaker if they meant what you are about to say. Use words such as "if I understand

[3] Note that while the scope of this paper is limited to the gap between systems and software engineering, there is a good probability that similar gaps exist between many engineering disciplines, so that the methodology described herein may be generally applicable at meetings of the integrated teams.

[4] No blood was shed in the process but we came close.

you, then …….", or "Do you mean….." This is the principle of applying feedback.

6. If, after you have rephrased what has been said and the person says, "No that's not it!" or the equivalent, then go back to step 4. You may need to invoke the STALL technique at this time (see below).

7. When the speaker finally agrees with you then you have (most probably) actually communicated and shared meaning.

In modifying active listening by the use of pattern matching, change step 5 to incorporate the pattern by adding words such as "this reminds me of the [Type A Scenario]", and "isn't this similar to [Type B]" and explain why you find a similarity in the current situation.

During the conversation use the STALL approach (Kasser 2001) to regulate matters[5]. STALL is an acronym for

Stay calm
Think
Ask questions and analyze
Listen
Listen

SUMMARY

Active listening is a well-established technique for bridging communications problems and sharing meaning. This paper has proposed that enhancing active listening with pattern matching may bridge some of the barriers to communications between systems and software engineers.

REFERENCES

Kasser J.E., Shoshany S., "Systems Engineers are from Mars, Software Engineers are from Venus", Thirteenth International Conference "Software & Systems Engineering & their Applications, Paris, December 2000.

Kasser J.E., "STALL", course notes from Software Engineering Project Management, UniSA, 2001.

BIOGRAPHY

Joseph Kasser D.Sc. C.Eng, CM, has been a practicing systems engineer for 30 years. He is the author of "*Applying Total Quality Management to Systems Engineering*" published by Artech House. Dr. Kasser is both a DSTO Associate Research Professor at the University of South Australia (UniSA) and a Distance Education Fellow in the University System of Maryland. He performs research into improving the acquisition process. Prior to taking up his position at UniSA, he was a Director of Information and Technical Studies at the Graduate School of Management and Technology at University of Maryland University College. There, he developed and was responsible for the Master of Software Engineering degree and the Software Development Management track of the Master of Science in Computer Systems Management (CSMN) degree. He is a recipient of NASA's Manned Space Flight Awareness Award for quality and technical excellence (Silver Snoopy), for performing and directing systems engineering. Dr. Kasser also teaches systems and software engineering via distance education. British born, he lived in the United States of America long enough to become sensitized to barriers to communications. He is currently learning his third dialect of English, namely Australian.

Sharon Shoshany M.Sc. (Electrical Engineering) has been a software engineer for 16 years, developing and managing the development of large real time systems in the defense domain. Her background in electrical engineering enables her to attempt to cross the gap to systems engineering or at least be able to appreciate its width.

[5] Stalling is a good way to deal with most problem situations.