The competitiveness of development organizations depends increasingly on the ability to improve their design and development processes faster than their competitors. A promising approach is to move our focus from engineering single systems to engineering families of systems by identifying “reusable” solutions that support the future development of multiple systems. An organization that focuses on a “range of similar” products can potentially take advantage of economies of scope, a benefit that comes from developing one asset used in multiple contexts, thus reducing development, maintenance, support and marketing costs, since products would share a common design, development effort, and actual components. Building systems from a common component base reduces risk and improves quality by using trusted, proven components; it reduces cycle time and cost of new applications by eliminating redundancy.

An asset-based approach allows the management of legacy systems more efficiently, increasing the likelihood of longer time-IN-market. And finally, an organization based on product lines evolves a common marketing strategy, and strengthens core competency around strategic business interests and goals. Thus, the overall goal is to produce quality products consistently and predictably by moving toward an asset-supported, component-based product lines development approach. This transition must be planned carefully encompassing not only technical issues but also organizational and business aspects.

Assessing the state-of-the-art helps in focusing the research agenda in the critical problems limiting the advancement of the field. What are the top remaining problems for SPL research? What are current top areas of SPL research? In this presentation, we address these and other questions referring to the technological challenges of building product lines. Due to space constraints however, we do not discuss specific technology in detail. Rather, we present a roadmap for the various research areas, relevance and promising approaches. We discuss briefly some of the most relevant contributions in the SPL research timeline from details of a survey we have conducted. We conclude with a discussion of the effort that lies ahead.
Symbolic Tracing for Program Reasoning

Joxan Jaffār
National University of Singapore
Singapore
joxan@comp.nus.edu.sg

Formal reasoning about programs is a key component of software engineering. Unfortunately, present tools for program reasoning are typically limited in scope, or else they require manual aid of an expert nature. In this talk, I will present a new framework for the generation of symbolic execution traces with emphases on transparency, flexibility and increasingly, practical performance. At the heart of the framework is Constraint Logic Programming (CLP), which was originally designed as a high-level declarative programming language. In this work, we adapt and extend the concepts of CLP to focus on representing the operational behavior of programs. We then show a number of algorithms that form the basis of efficient implementation.
Paradox between Software Development Productivity and Success Rate
Are we striking the right balance?

Nirbhaya Pathak
Global Technology Development Group
Motorola Technology Sdn Bhd, Penang, Malaysia
a17385@motorola.com

Software systems produced in recent years are more complex and sophisticated than those of yester years. At the same time there is constant pressure to produce quality software in a shrunk cycle time due to fierce competition and rapid change in technology. Software industry in general is struggling to strike the right balance between productivity and success rate. The success rate is defined from two schools of thoughts: “Does the delivered system meet or exceed the expectations we had for it?” and “Did the system come in "up to spec," on time, and on budget?”

The Gartner Group reported that $3.7 billion was spent worldwide in 2004 on application development tools. This was up more than 5 percent from that spent in 2003; spending is increasing progressively year after year, but the success rate has not shown results in proportion to spending as shown below (Table 1) through a study conducted by “Standish’s Group CHAOS report”.

Table 1: Software project success rates reported by Standish Group

<table>
<thead>
<tr>
<th>Date</th>
<th>Success Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>First CHAOS report</td>
<td>1994</td>
</tr>
<tr>
<td>&quot;Extreme CHAOS&quot;</td>
<td>2001</td>
</tr>
<tr>
<td>Most recent CHAOS</td>
<td>2003</td>
</tr>
</tbody>
</table>

Data indicate that the improvement is about 1.7% a year and appears to be linear based on this small sample of data. If the current improvement rate continues, we should achieve a 50 percent success rate in the year 2014. Indicators demonstrate that success rate has been creeping up in recent years but there has been NO breakthrough.

There has been significant amount of investment poured in two areas: “Tools” and “Methodologies”. This investment indeed has improved the productivity but the question to ask “Why does the needle on success rates move so slowly?” It is almost as though all the advances in methodologies and tools have helped us build more ambitious systems, yet there is some other factor that is keeping us from being successful more of the time. Perhaps there is something in the approach of the past several decades that is missing the point.

This keynote describes an approach to achieve continuous improvement in success rate and productivity by using the combination of “Methodologies”, “Processes” and “Changing behavior pattern of software Engineers”. Undoubtedly there has been substantial amount of work in “Methodologies” and “Processes” over the years, but amalgamation of two in a way so that engineers who are developing the software can relate to and feel motivated to improve success rate and productivity, has been missing. This keynote addresses this issue, by focusing on the following:

- Systematic inclusion of practices of *Six Sigma for Product Development (SSPD)* to translate the Voice of Customer (VOC) into measurable Requirements. Historical data have shown that errors in requirements contribute to a consistent, absolute project failure rate (~50%) that can’t be reduced.
- Critical KPIs from CMMi (QPM and PP) to ensure project life cycle remains predictable.
- Use of “*Agile*” methodology for development to ensure motivation of engineers remains high at the same time generating better productivity and low defect density in turn resulting into high probability of success rate.