Khrishna (2011) fala que, programar é uma habilidade que pode ser aprendida de forma semelhante a uma matéria escolar como matemática, a prática faz com que o programador fique cada vez melhor. Assim como na matemática 1+2 sempre será igual a 3, na programação a instrução ou está certa, ou está errada. Não há outra solução possível.  A autora alerta que já o desenvolvimento de sistemas é algo um tanto mais complicado. Desenvolver software é entender seus clientes e usuários, conversar com eles, observá-los, trabalhar em equipe e gerar um produto de qualidade. Não há necessariamente a resposta certa. Prever comportamento não é possível.

Esse capítulo abordará algumas facilidades e dificuldades no desenvolvimento de software tradicional.

A quantidade de linguagens de programação disponíveis. As linguagens de programação são as principais ferramentas de desenvolvimento de software. O site Scriptol (2012) contabiliza mais de 400 linguagens de programação disponíveis para serem utilizadas nas mais variados tipos de problemas. Desde as linguagens mais básicas até aquelas utilizadas para manipular Big Data e sistemas críticos.

O reuso de partes de outros sistemas. Os especialistas em desenvolvimento de software sabem que não criam um sistema completo, do zero. Ao contrário eles reutilizam suas soluções que funcionaram no passado e que pode ser aproveitada no novo software (GAMMA et al., 1994, p. 11).

O custo decrescente de hardware o que possibilita que mais e mais pessoas tenham equipamentos eletrônicos e aumenta a demanda de software. Uma tabela comparativa desenhada pela Software Metrics (2012) apresenta o comparativo entre o índice de custo de hardware e a estimativa de softwares desenvolvidos por ano.

| **Ano** | **Índice do Custo de Hardware** | **Quantidade de Softwares** |
| --- | --- | --- |
| 1960 | 678 |  |
| 1970 | 75 |  |
| 1980 | 10,5 | 150 |
| 1990 | 0,5 | 19.000 |
| 2000 | 0,3 | 42.000 |

Tabela Estimativa de custo de hardware e quantidade de softwares

Os desenvolvedores de software são classificados como profissionais de conhecimento (do inglês (knowledge worker) e como tal são motivados não só pelo salário, mas principalmente conquistas e capacidade de resolver problemas (AMABILE et al., 2010). Visão corroborada por Asproni (2004, p. 3) que classifica como o item motivacional mais importante para esses profissionais a conquista, a possibilidade de crescimento na carreira e o trabalho em si. Nas pesquisas realizadas o salário aparece apenas na décima posição.

## Dificuldades

Porém, nem tudo são flores no processo de desenvolvimento de softwares, há grandes dificuldades que devem ser superadas. Dentre elas, algumas das mais importantes são: Restrições inerentes a projetos, a dificuldade de comunicação, complexidade dos softwares atuais, redução de custos de TI, necessidade de conformidade de padrões de mercado.

As restrições inerentes ao projeto são definidas pelo GP3 (2012) como um conjunto de fatores que tomam grande parte do tempo do gerente e da equipe do projeto. Eles chamam o conjunto dessas restrições de “restrição tripla” — escopo, tempo e custo do projeto — de necessidades conflitantes do projeto. Considera-se ainda a qualidade do projeto como uma quarta variável, a qual é afetada pelo balanceamento das três demais. O site afirma que “Existe um equilíbrio natural entre essas quatro ‘forças’ dentro do projeto, que é estabelecido no momento em que as linhas de base de escopo, tempo e custo são acordadas entre as partes envolvidas na realização do projeto. A partir desse ponto, mudanças em uma dimensão (por exemplo, escopo) serão refletidas em uma ou mais entre as três outras” (GP3, 2012).

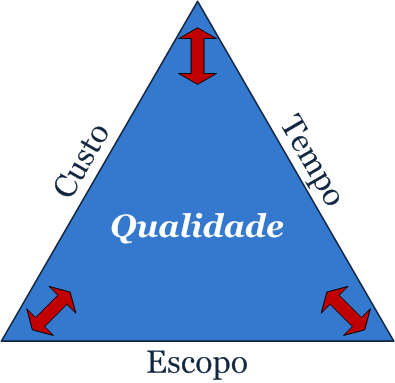


Figura Triângulo das restrições de projetos

O PMI (2008, p.6) em seu famoso livro de conhecimento PMBOK vai além e diz que os projetos sofrem pelo menos as seguintes restrições: Escopo, custo, tempo, qualidade, risco e recursos. A renomada instituição afirma ainda que esses não são as únicas restrições possíveis podendo existir ainda mais.

Boehm corrobora com a informação e diz: “No processo de criação de software (e em todo mundo corporativo) nós lidamos com restrição de recursos. Nunca há tempo ou dinheiro suficiente para cobrir todas as funcionalidades que são desejadas no produto de software” (BOEHM, 1984, p.1).

Klawe (KLAWE et al., 2005, p.27) ressalta a restrição de recursos humanos. Cada vez há menos profissionais disponíveis. O mesmo diz prof. de Stanford Mehran (2012), apenas 6% dos alunos da graduação da ilustre universidade que iniciam o curso de Computação Científica irão tornar-se “engenheiros” de software.

Além das restrições de projetos, outra grande dificuldade é a comunicação. Ela é necessária para estabelecer confiança entre a equipe de desenvolvimento e seus clientes, para obtenção de requisito, feedback, negociação, planejamento, auxiliar na execução entre outros, porém embora seja algo primordial, não é simples de ser estabelecida. Passadori (2003) evidencia alguns fatores que atrapalham a comunicação: Desorganização de ideias, vícios de linguagem, dificuldade com o vocabulário, inadequação de recursos audiovisuais, prolixidade, excessiva objetividade, arrogância e prepotência e excesso de humildade. As consequências geradas pela baixa comunicação podem ser falha no levantamento de requisitos (BLASCHEK, 2012, p.1), informação desnecessária, criação de “silos de informação”, interrupções, informações que chegam desatualizadas, reuniões sem foco (GRIMES, 2009). Para (DOWN, 2008, p. xxix) 90% das razões de falha de um projeto podem ser representadas por problemas de comunicação. Cockburn (2001, p. 84) apresenta um gráfico em que contraria a ideia básica de uma boa comunicação. Para ele a comunicação escrita possui uma eficiência muito baixa e a comunicação frente a frente, é a mais eficiente.

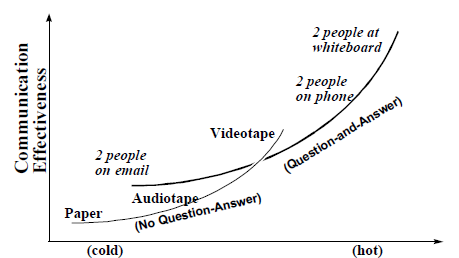


Figura Gráfico de Eficiência da Comunicação de Cockburn

Outra grande dificuldade é a complexidade dos softwares.

# Bibliografia

AMABILE, Teresa M. KRAMER, Steven J. The HBR List: Breakthrough Ideas for 2010. Harvard Business Review. In: http://hbr.org/2010/01/the-hbr-list-breakthrough-ideas-for-2010/ar/1 . Acessado em dezembro de 2012: Publicado em 2010.

COCKBURN. Alistar. Agile Software Development. Addison-Wesley Professional. Boston – USA: 2001.

DOWN, Willian. TAYLOR, Bruce. Project Management Communications Bible. Wiley Publishing, inc. 1st edition. Indianapolis – USA: 2008.

GAMMA, Erich. HELM, Richard. JOHNSON, Ralph. VISSIDES, John. Design Patterns: Elements of Reusable Object-Oriented Software. Ed. Addison-Wesley: 1994.

GRIMES, Andrew. Five dangers of poor project communication. In: <http://www.semantico.com/2009/11/five-dangers-of-poor-project-communication/>. Acessado em dezembro de 2012. Publicado em 2009.

PASSADORI, Reinaldo. Problemas mais comuns de comunicação. In: <http://www.consultores.com.br/artigos.asp?cod_artigo=244>. Acessado em dezembro de 2012. Publicado em outubro de 2003.

PMI, Project Management Institute. PMBOK – Project Management Book of Knowledge. 4th edition. PMI. Pennsylvania – USA: 2008.

KRISHNA. Programming is Easy, Software Development is Hard. In: <http://www.thoughtclusters.com/2011/01/programming-is-easy-software-development-is-hard/#comment-22034>. Acessado em dezembro de 2012: Publicado em 2011

SCRIPTOL. List of Programming Languages in Alphabetical Order. In: <http://www.scriptol.com/programming/list-programming-languages.php>. Acessado em dezembro de 2012.

SOFTWARE METRICS. Hardware Costs and Software Development. In: <http://www.softwaremetrics.com/Articles/HardwareandSoftware.htm>. Acessado em dezembro de 2012.