

# UM MODELO HELICOIDAL DO CICLO DE VIDA PARA GESTÃO DE PROJETOS DE SISTEMAS DE TECNOLOGIA DA INFORMAÇÃO

Antonio Carlos Pinto Dias Alves (COPPE / UFRJ)  
acpalves@bb.com.br

## **Resumo**

*Nos últimos tempos temos visto um grande avanço nas metodologias de desenvolvimento de sistemas computacionais. No campo das aplicações orientadas a objetos, por exemplo, a metodologia UML vem se firmando como a principal ferramenta de auxílio ao desenvolvimento de sistemas. Ao mesmo tempo, nota-se como sendo cada vez menor a importância atribuída a uma ferramenta outrora de grande valor, o ciclo de vida do sistema.*

*Existem várias razões para isso. Alguns apontarão que os modelos de ciclo de vida são ferramentas obsoletas, que tiveram sua utilidade para o acompanhamento do desenvolvimento de sistemas seqüenciais até o tempo do aparecimento das técnicas ditas estruturadas, quando alguns pesquisadores, como Edward Yourdon, estabeleceram modelos de ciclos de vida estruturados. Quando as ferramentas para o desenvolvimento de sistemas orientados a objetos começaram a ganhar campo, o conceito de ciclo de vida começou a ser relegado a um plano inferior e dentre as metodologias de desenvolvimento surgidas nenhuma realmente incorporou o conceito. Mas provavelmente a razão mais forte para esse ostracismo é que os modelos de ciclos de vida existentes na literatura não fornecem uma possibilidade efetiva de medições quantitativas sobre o sistema sob desenvolvimento.*

*Mesmo no modelo mais completo do ciclo de vida espiral, as iterações espirais podem ter qualquer tipo de interpretação e são de pouca utilidade para os gerentes de projeto, que precisam lidar com prazos e custos determinados pela alta direção. Ainda, o ciclo de vida espiral pode ser de pouca valia em alguns tipos de projeto e não é possível uma adaptação direta. Dessa forma, o uso de ferramentas automatizadas ganhou impulso pelas necessidades de rápido desenvolvimento, custos reduzidos e erros mínimos sem, contudo, conseguir-se uma visão total dos custos e prazos*

*em cada fase do andamento do projeto, desde a fase inicial de análise até o teste integrado do sistema.*

*Este trabalho apresenta uma ferramenta, o modelo de ciclo de vida helicoidal, que não é uma mera evolução do modelo espiral. Neste modelo de ciclo de vida, são possíveis medições precisas de prazos e custos, bem como dos estágios de desenvolvimento do sistema. A aplicação prática dessa ferramenta no desenvolvimento de vários sistemas mostrou sua utilidade ao permitir não só métricas precisas como corte de custos que, de outra forma, não seriam possíveis. O ciclo de vida helicoidal representa o modelo mais geral que um ciclo de vida computacional pode ter e assim pode ser facilmente reduzido a qualquer um dos ciclos de vida tradicionais e mesmo a metodologias orientadas a ob*

### **Abstract**

*In recent years we have been seeing great advances in methodologies for the development of computational systems. For example, in the field of object oriented applications, UML methodology have been setting a standard as an auxiliary tool for systems development. At the same time, one can note as losing importance a tool once given great value, the system life cycle.*

*There are many reasons for that. Some would point that models for life cycle are obsolete tools, that had usefulness for accompanying the development of sequential systems up to the appearance of the so called structured techniques, when some researchers, as Edward Yourdon, established structured life cycle models. When the tools for development of object-oriented systems rose, the concept of life cycle started being neglect to a lower plane and, among the emerging development methodologies, none had really incorporated that concept. On the other hand, the strongest reason for this ostracism is that the existing life cycle models in literature do not allow an effective possibility of quantitative measurements over the system being developed.*

*Even in the most complete model, the spiral life cycle, the spiral iterations can be given any kind of interpretation (or misinterpretation) and are of little utility for project managers, who need to deal with terms and costs determined by the high administration. Also, the spiral life cycle can be worthless in some kinds of projects and any direct adaptation is impossible. This way, the use of automated tools grew up by the need of fast development, reduced costs and minimum errors without, however, bring a global vision of the costs and terms in each phase of the project, since the initial analysis up to the integrated test of the system.*

*This work presents a tool, the helicoidal life cycle model, which is not a mere evolution of the spiral model. With this life cycle model, they are possible precise measurements of terms and costs as well as the system development stages. The practical application of this tool in the*

*development of various systems showed its utility in allowing not only precise measurements but also cost reductions that, in any other way, would not be possible. The helicoidal life cycle represents the most general model a life cycle can achieve and, this way, can be easily reduced to any of the traditional life cycles or those that represent object-oriented methodologies, like the Schlaer-Mellor. Finally, the helicoidal life cycle model surely will be the model from which best automated tools will arise, by its qualities of automatic incorporation of quantitati*

*Palavras-chaves: Ciclo de Vida, Sistema, Tecnologia da Informação, Software*

## 1. Introdução

Os ciclos de vida tradicionais tais como cascata, prototipagem e espiral; os modelos de ciclo de vida estruturados; e os ciclos de vida orientados a objeto (como o Schlaer-Mellor), todos tem o objetivo primordial de permitir um gerenciamento de alto nível do estado de desenvolvimento de projetos computacionais em geral e, no foco deste trabalho, de projetos de tecnologia da informação (TI).

O mais antigo, o “ciclo em cascata” [1][2][3], ainda que esteja perdendo importância, pode ainda ter alguma utilidade, especialmente em sistemas críticos de segurança e/ou quando são necessárias revisões em sistemas antigos (este caso ocorreu alguns anos atrás quando do bug Y2K). A prototipagem [1][2] é um princípio evolucionário para a estruturação do ciclo de vida. Os incrementos ao sistema são progressivamente entregues ao usuário conforme vão sendo desenvolvidos e, dessa forma, uma versão inicial (ou protótipo) vai sendo progressivamente transformada na versão final. De forma geral, a prototipagem é vista mais como uma ferramenta de auxílio ao processo de compreensão das necessidades do usuário. O modelo de ciclo de vida tradicional mais “moderno” é o “ciclo em espiral” [1][2][4] e foi desenvolvido de forma a permitir um gerenciamento mais efetivo do projeto, incluindo as previsões das várias iterações pelas quais passam projetos de media a alta complexidade (de 100.000 a mais de 1.000.000 de linhas de código).

Posteriormente, alguns pesquisadores propuseram o ciclo de vida estruturado [5] que foi desenhado para ser um instrumento de controle em projetos que utilizam técnicas de desenvolvimento estruturadas. Conforme as linguagens e técnicas orientadas a objeto evoluíram, a teoria do desenvolvimento orientado a objetos apareceu. Este tipo de desenvolvimento primeiro modela o problema por inteiro, ao invés de permitir linhas e linhas de código indiscriminado. Em geral, essa abordagem necessita mais tempo de análise mas, como aborda o processo por inteiro, um modelo bem aplicado pode resolver tanto o problema original como outros que apareçam durante o processo de desenvolvimento. Uma metodologia, a UP (Unified Process) usa a UML para implementar suas abordagens metodológicas e inclui seu próprio modelo de ciclo de vida [6][7].

Entre os vários métodos de desenvolvimento orientados a objetos, uma categoria inclui, entre outros, os métodos Booch, Rumbaugh e OMT. Essa abordagem faz iterações por meio de análise, projeto e codificação e incluem informações adicionais em cada passagem até que a iteração final esteja em nível de código. Uma outra categoria inclui um modelo de ciclo de vida muito bem sucedido, o Schlaer-Mellor, que automaticamente interpreta e traduz modelos de análises orientadas a objeto em código objeto, tendo por base informações passadas por ferramentas CASE. O Schlaer-Mellor usa uma notação concisa para implementar a tradução automática de métodos em código objeto e particiona o problema em domínios lógicos, dessa forma possibilitando a oportunidade de reuso de código em nível de domínio. Os benefícios do reuso são especialmente úteis em projetos de larga escala e/ou longos prazos porque antecipa atualizações ou futuros projetos relacionados [7].

De volta aos modelos tradicionais, a *análise de sistemas* faz, então, uma análise do problema. A seguir, o *projeto de sistemas* cria um projeto que implementa essa análise e o código é então escrito de forma a implementar o projeto. O teste de código pode ou não descobrir bugs, o

projeto em si pode mudar (necessitando que o código seja refeito) ou mesmo o sistema inteiro pode mudar, devido a novas análises, que mudarão o projeto, que necessitará de novo código. Contudo, como sabemos, muitas vezes ocorrem modificações no código em virtude de novas tecnologias sem que a análise ou o projeto necessitem de mudanças.

No Schlaer-Mellor a análise de domínios independentes ocorre, em geral, tanto paralela como concorrentemente. Tanto as simulações como os testes podem ocorrer no nível de análise. Com o uso de ferramentas adequadas o modelo pode transformar os projetos diretamente em código. Caso ocorram modificações em um domínio, essas modificações não necessariamente afetam outros domínios. Em geral, as fases de análise e codificação permanecem em sincronismo. De qualquer forma, como pode alguém, *quantitativamente*, ver o sincronismo entre as fases ?

## 2. O problema das métricas

Sempre houve tentativas de fazer a formalização das diferentes fases nos antigos modelos de ciclo de vida. Essas formalizações definiam os documentos a serem emitidos, as fases a serem cumpridas e, em alguns casos, as interações e iterações a serem seguidas. Infelizmente, informações de grande importância, as métricas, nunca puderam ser explicitamente incorporadas em nenhum deles.

Por exemplo, nos modelos de ciclo de vida tais como o cascata, a prototipagem e o estruturado as métricas não são nem mesmo contempladas. Uma primeira tentativa foi feita com o ciclo de vida espiral em que o aumento do raio indica o quanto se está próximo de completar o sistema. Contudo, os eixos que são definidos nesse modelo não tem qualquer significado relevante, tendo apenas o objetivo de separar os diferentes quadrantes. O eixo vertical, que pretende mostrar os custos cumulativos do projeto, na verdade nada mostra, visto que assume valores cíclicos positivos e negativos.

No Schlaer-Mellor, existem passos e marcas definidos para rastrear o progresso das fases do projeto quando, na verdade, deveriam haver marcas percentuais do progresso daquelas fases. Em um projeto orientado a objetos, é possível estimar o numero de objetos necessários a partir do numero de entidades (táteis ou não táteis) com as quais o projeto tem que lidar. De qualquer forma, mesmo o Schlaer-Mellor não permite a inclusão de métricas no ciclo de vida e, dessa forma, ele não contribui nem para uma gestão efetiva nem para uma análise gerencial quantitativa.

## 3. O ciclo de vida helicoidal

No ciclo de vida helicoidal o estado de desenvolvimento de um sistema é precisamente definido por um ponto **P** no espaço definido por um sistema de coordenadas cilíndricas, em que  $P = f(R, \theta, T)$  como pode ser visto na figura 1(a). As dimensões  $R, \theta, T$  são assim definidas:  $R$  representa a quantidade de recursos alocados para a fase de desenvolvimento em consideração (análise, projeto, codificação...), por exemplo, pessoal, valores monetários, recursos computacionais, etc. As fases são divididas em ciclos chamados *iterações*. A coordenada  $\theta$  está relacionada com a porcentagem já cumprida de uma iteração. Os quadrantes, que são percorridos

no sentido horário (tal como no modelo espiral), podem ser associados, cada um, a uma porcentagem de completude da fase. (por exemplo, 25% de completude caso se considere que um ciclo deve estar completo quando  $\theta$  alcança  $360^\circ$ ). Alternativamente, cada  $360^\circ$  percorridos em  $\theta$  podem ser associados a um dos quadrantes do modelo espiral (por exemplo, o estágio de avaliação pelo usuário). Talvez a maneira que melhor permita o uso integral da flexibilidade permitida pelo modelo helicoidal seja aquela mostrada na figura 1(b). Tal qual na prototipagem, divide-se cada iteração em fatias, como uma pizza. Chamaremos cada fatia de um *setor*. Cada setor é definido por um ângulo  $\theta$  arbitrário e define um *estágio*. Na figura 1(b) estão desenhados 5 estágios do primeiro ciclo da fase de análise. O primeiro, o setor 1, define o estágio de *coleta de informações das necessidades do usuário*. No Segundo estágio é realizada a *análise de risco* de todo o projeto. Os terceiro e quarto setores mostram que as porcentagens da iteração associadas aos estágios de planejamento e engenharia são maiores do que para as anteriores. Finalmente, a amplitude do setor 5 pode mostrar a porcentagem designada para o estágio de *avaliação pelo usuário*. Certamente as amplitudes dos ângulos que definem cada setor (bem como as próprias designações dos estágios) são livremente definidas pelas necessidades do projeto.

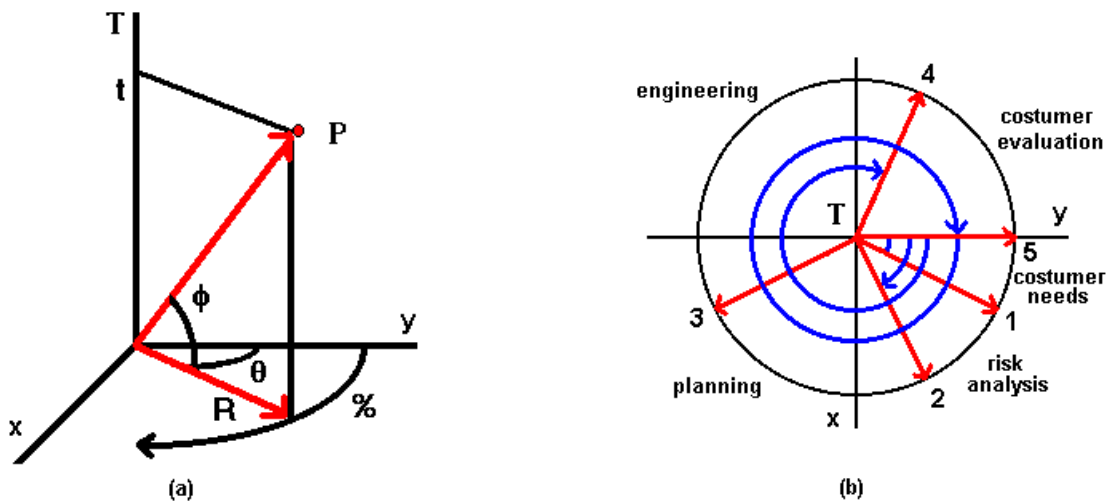
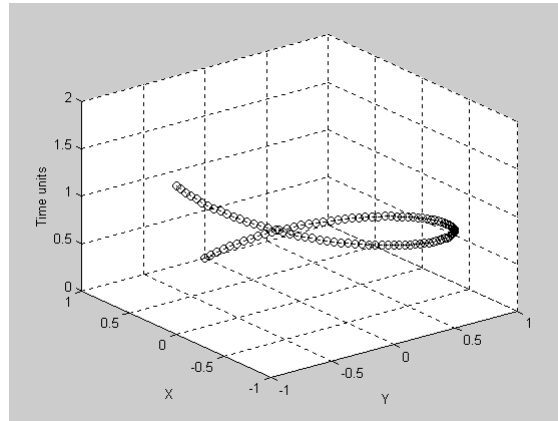


Figura 1 – (a) Espaço de coordenadas cilíndricas e (b) setores e estágios do ciclo de vida helicoidal.

A variável  $T$  é a dimensão temporal do ciclo. Quanto maior for, mais tempo foi gasto para um determinado estágio ser completado. Tal como no ciclo espiral, onde cada volta da elipse significa que o desenvolvimento do sistema está caminhando para seu fim, quanto maior o valor de  $T$ , isso deve indicar que o projeto do sistema está cada vez mais chegando ao fim. Nesse espaço de coordenadas cilíndricas um dado ponto  $P$  percorre um caminho helicoidal, como pode ser visto na figura 2. Pode-se notar que, se o valor de “ $T$ ” associado a um determinado estágio for muito grande, então esse estágio está, provavelmente, consumindo muito tempo (e dinheiro).



**Figura 2** – Caminho helicoidal percorrido por um ponto “P” em um espaço de coordenadas cilíndricas. Cada vez que o ponto completa 360° no plano xy, ele perfaz uma *iteração*.

As dimensões são independentes. Assim, por exemplo, mais ou menos recursos podem ser alocados (variando **R**) sem afetar as outras dimensões. Por outro lado, ao se variar a alocação de recursos, as dimensões  $\theta$  e **T** devem variar com o progresso do desenvolvimento do sistema. Claro que as marcações da dimensão **T** devem ter espaçamento constante e, assim, apenas a inclinação da hélice irá variar, como será visto mais adiante. Tal como no modelo espiral, os eixos **X** e **Y** não tem um significado importante e agem como principalmente separadores dos quadrantes. Um uso possível ocorre quando alguém deseje descrever as coordenadas do ponto **P** por meio de suas equações paramétricas.

As figuras 3(a) e (b) mostram um exemplo de uso. As iterações plotadas são para a fase de análise. Desde o início do projeto até o ponto T1, os recursos alocados somam R1. Nesse ponto, novos recursos são alocados (por exemplo, mais um analista) e o tamanho do raio muda para R2. As outras dimensões não são afetadas. Um caso mais real é o visto nas figuras 4(a) e (b). A alocação de novos recursos faz com que a fase de análise se desenvolva mais rapidamente e isso é visualizado graficamente pelo menor valor do incremento que é dado à dimensão **T**. Neste novo modelo de ciclo de vida as diferentes fases do desenvolvimento são plotadas de uma forma muito adequada e podem ser facilmente controladas, facilitando a gestão do projeto, como mostra a figura 5.

O ciclo de vida helicoidal permite significativa economia de tempo e recursos porque, diferentemente de todos os demais tipos de ciclo de vida, neste modelo as diferentes fases do desenvolvimento podem se superpor. Chamamos essa superposição de *interação*.

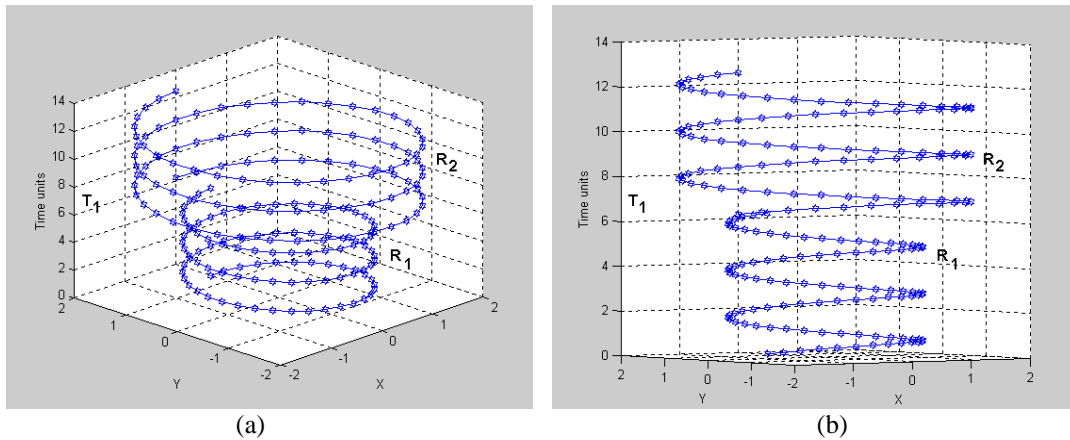


Figura 3 – Adicionando recursos a um projeto. Vistas (a) 3D e (b) Frontal.

A figura 6 mostra três fases do desenvolvimento de um sistema: a fase de análise é a primeira a começar e as fases de projeto e codificação residem dentro e fora da fase de análise, respectivamente. Como se vê, mesmo a fase de análise ainda ocorrendo, a fase de projeto pode se iniciar sem que ocorra qualquer perda do controle dos recursos. Também pode ser visto que a fase de codificação começa antes que as fases de análise e projeto terminem. Não apenas a superposição de fases é possível mas também ela é encorajada, como forma de economizar tempo e recursos (aliás, todos nós, analistas, sabemos que essa superposição sempre existiu na prática). Contudo, é praticamente impossível medir e controlar essa superposição com qualquer outro modelo de ciclo de vida. Como já vimos, quando duas ou mais fases se superpõem dizemos que existe uma *interação* entre elas.

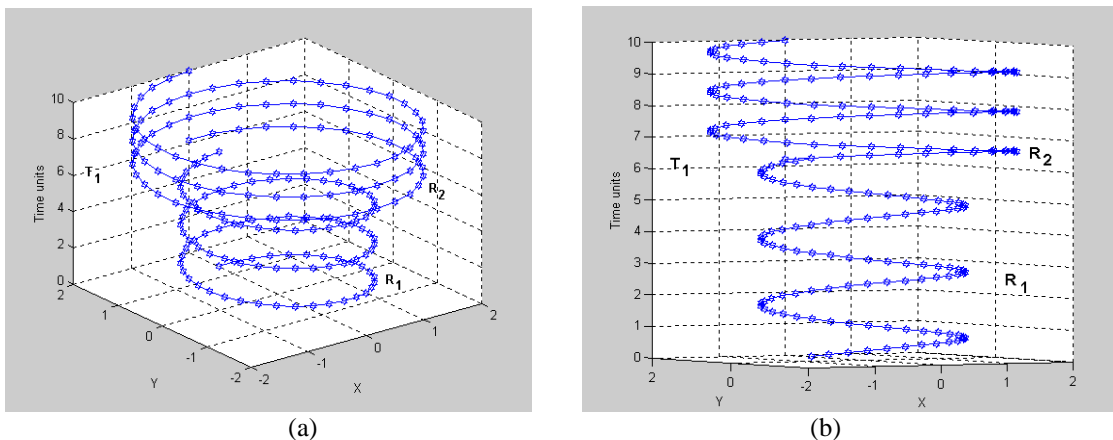
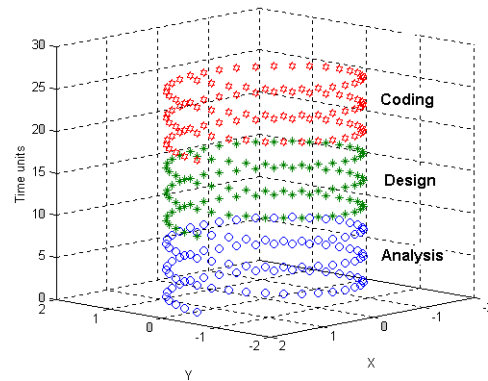


Figura 4 – Adicionar recursos a um projeto diminui o tempo de desenvolvimento e pode reduzir custos. Vistas (a) 3D e (b) Frontal.

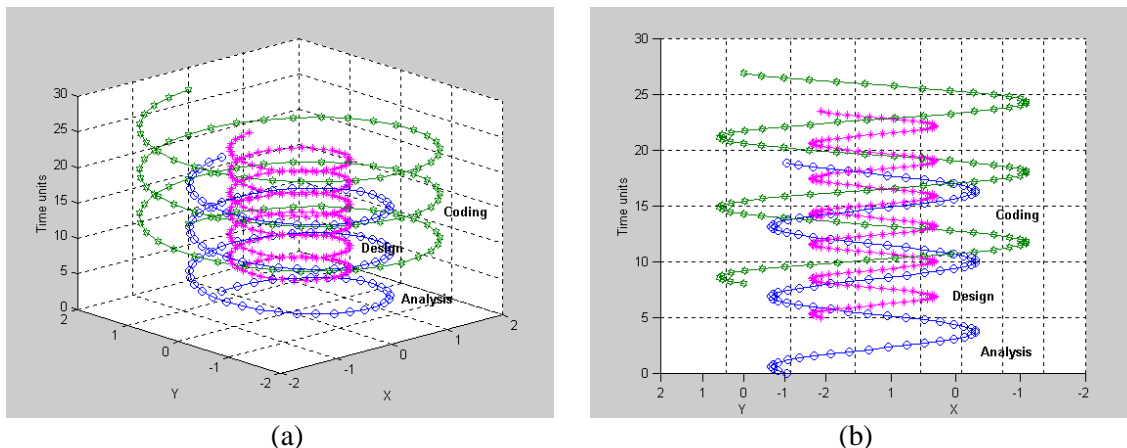
## 4. Exemplos

O ciclo de vida helicoidal já foi usado na prática de forma a gerir e controlar o desenvolvimento de alguns sistemas. Dois exemplos são aqui apresentados para comparação. Baseados em experiências prévias em projetos similares, a cada projeto foi associado um prazo para sua conclusão bem como o prazo e orçamento disponível para cada fase, dependendo do ciclo de vida que tenha sido usado como paradigma. Para cada um dos projetos foi designado um grupo composto por 3 analistas de sistemas e 3 analistas programadores.

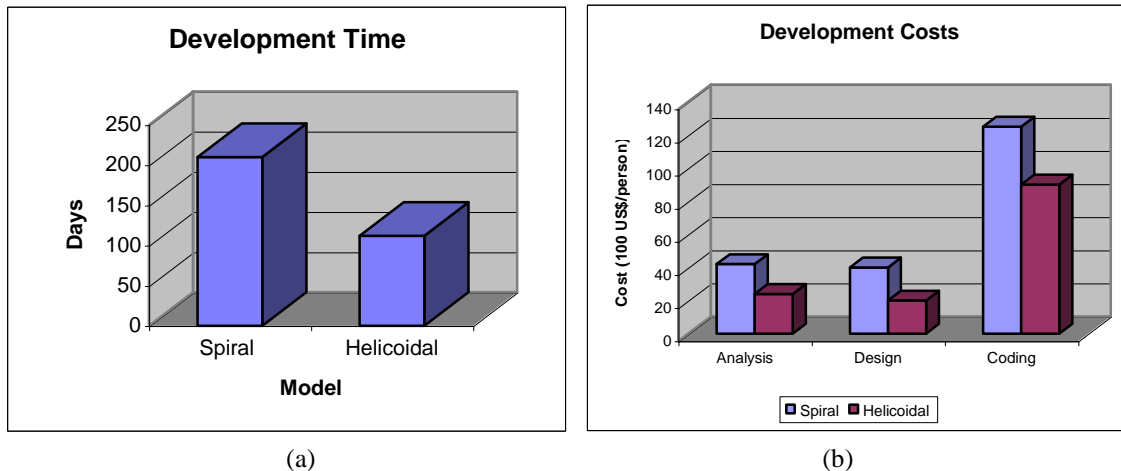


**Figura 5** – Diferentes fases do desenvolvimento mostradas em “cascata” no ciclo de vida helicoidal.

O primeiro projeto foi conduzido por uma empresa de consultoria e lidou com a tarefa de desenvolver um sistema para uma companhia de telemarketing; desde estender as interfaces com a companhia telefônica local e o PABX, a instalação em si da parafernália eletrônica até a instalação e configuração da rede de computadores, juntamente com o desenvolvimento e teste dos softwares necessários. O projeto seguiria o ciclo de vida espiral e, assim, os prazos e custos foram associados àquele tipo de ciclo de vida. Contudo, ao invés de seguir o modelo espiral, o gerente do projeto escolheu tentar o modelo helicoidal. A figura 7 mostra o resultado.



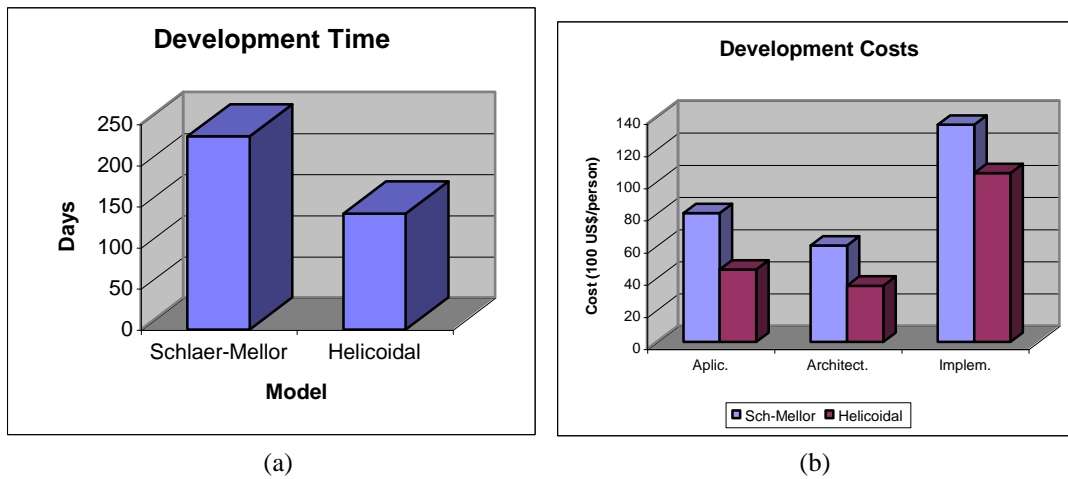
**Figure 6** – Interação de diferentes fases do desenvolvimento em vistas (a) 3D e (b) Frontal.



**Figura 7** – Ciclo helicoidal x Ciclo espiral. (a) Redução total no tempo de desenvolvimento e (b) Redução nos custos por fase.

Em (a) é mostrada a redução total no tempo de desenvolvimento do sistema contra o que seria esperado de um desenvolvimento com o modelo espiral. O uso do modelo helicoidal trouxe uma redução nos custos de desenvolvimento de quase 50%. A figura 7(b) mostra a redução dos custos em fases isoladas do processo. Conseguiu-se essa economia devido à deliberada interação entre as fases.

O segundo projeto lidou com a tarefa do desenvolvimento de um ambiente de simulação completo para a emulação e controle de alguns sistemas e condições de falha de uma usina nuclear [8]. O sistema demandou não apenas a programação do software como também um pesado hardware teve que ser construído. O ciclo Schlaer-Mellor foi então escolhido para controlar o projeto devido à sua característica de particionar o problema em domínios lógicos e então integrar o código gerado com a arquitetura disponível. Neste caso, contudo, decidiu-se usar o Schlaer-Mellor principalmente como ferramenta de metodologia e deixar o modelo helicoidal gerir o ciclo de vida do sistema. O resultado é mostrado na figura 8. Em (a) vemos que a redução total no tempo de desenvolvimento foi em torno de 30% abaixo do que havia sido estimado. Todas as análises dos domínios, fosse o de aplicação, o de arquitetura ou o de implementação, experimentaram reduções em seus prazos e custos.



**Figura 8** – Ciclo helicoidal x Modelo Schlaer-Mellor. (a) Redução total no tempo de desenvolvimento e (b) Redução dos custos por fase.

Para a análise do domínio de aplicação a redução de custos foi de quase 50%. De muitos estudos (que não são referenciados aqui) sabemos que quase 40% do tempo disponível para a equipe técnica é despendido em atividades não técnicas (tais como reuniões, tarefas administrativas, etc.). O ciclo de vida helicoidal pode rastrear eficientemente em que ponto(s) tempo e dinheiro estão sendo desperdiçados dessa forma promovendo uma gestão eficiente da alocação dos recursos disponíveis. A porcentagem de erros nas fases de projeto e codificação foi, em geral, a que era prevista sem interações entre as fases, mostrando que a interação controlada (superposição) de fases não contribui para um aumento na quantidade de erros em projeto e/ou codificação.

## 5. Redução aos ciclos de vida convencionais

O ciclo de vida helicoidal pode ser facilmente reduzido a qualquer um dos ciclos de vida tradicionais. A figura 9 mostra como, ao não se permitir interações entre as fases e simplesmente invertendo o diagrama, o ciclo helicoidal pode ser visto como o em cascata. A redução ao modelo de prototipagem também é possível e pode ser visualizada na figura 10. Este é um caso bastante interessante. Como a prototipagem já divide o ciclo em setores, a analogia é imediata. Como já foi dito, cada setor define um estágio. Assim, as iterações do modelo helicoidal são agrupadas de forma que um observador colocado diretamente sobre elas veria apenas uma única iteração. Esta iteração solitária é o próprio ciclo de prototipagem.

Os setores permanecem definindo os estágios e a variável “T” continua medindo o tempo gasto em cada estágio, permitindo o controle de prazos e custos. Uma outra construção é possível. Cada setor do ciclo de prototipagem pode ser associado a uma iteração do modelo helicoidal. O controle destas iterações leva a um controle ótimo de tempo e recursos.

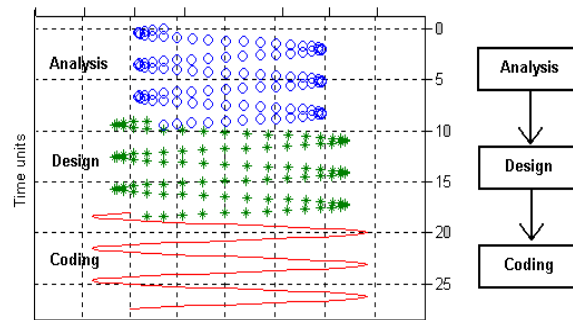


Figura 9 – Redução do ciclo de vida helicoidal ao ciclo “cascata”.

A redução ao modelo espiral também é imediata. Cada iteração do modelo helicoidal pode ser imediatamente transcrita para o espiral fazendo-se o ajuste da taxa de crescimento da dimensão radial do ciclo espiral para uma razão de crescimento de T no ciclo helicoidal (e vice-versa).

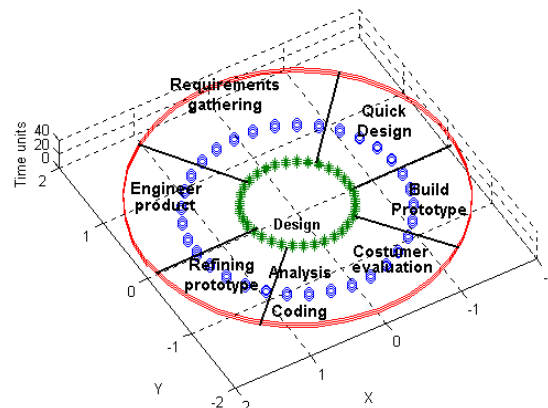


Figura 10 – Redução do ciclo helicoidal ao de prototipagem.

Em qualquer dos casos de redução alguma informação é perdida. Por exemplo, nos casos em que a redução é feita para o modelo espiral, perde-se a visualização da alocação de recursos devido à supressão da dimensão radial. A perda de informação é ainda maior no caso de redução ao modelo de prototipagem e será severa se a redução for feita ao modelo cascata.

O modelo de ciclo de vida estruturado tal como foi proposto por Yourdon [5] é, tanto quanto sabemos, uma adaptação do modelo cascata para ser usado com técnicas estruturadas. Dessa forma, a redução do modelo helicoidal para o estruturado é feito da mesma forma que para o modelo cascata. A redução do modelo helicoidal para o Schlaer-Mellor também pode ser feita de uma forma adequada. No Schlaer-Mellor as diferentes análises de domínio rodam umas dentro das outras, dependendo dos recursos alocados a cada uma.

## 6. Conclusões

Pelas considerações apresentadas, pode-se ver que o modelo de ciclo de vida helicoidal representa convenientemente as diferentes fases e estágios do desenvolvimento de software. Comparado aos ciclos de vida tradicionais, ele pode rastrear os prazos e os custos de um projeto de uma forma bastante adequada. Vimos que mesmo o ciclo de vida espiral, em sua tentativa de rastrear custos pode levar a más interpretações devido a seu sistema de eixos, o qual carece de sentido. Por sua vez, o modelo de ciclo helicoidal mostra dimensões que tem significados bem definidos, dessa forma permitindo um controle preciso dos recursos sendo usados e os progressos atingidos.

Ainda mais, o Ciclo de Vida Helicoidal permite a superposição das diferentes fases do desenvolvimento, promovendo assim economia de tempo e recursos. Ainda que algumas fases possam ser completamente envolvidas por outras, cada uma pode ser rastreada independentemente enquanto que o desenvolvimento do sistema completo pode ser rastreado por um outro ciclo de vida helicoidal que cobre os demais em uma forma de “casaca de cebola”. Também, pelos exemplos de uso apresentados, vimos que a redução nos custos e no tempo de desenvolvimento não podem ser ignorados. Em alguns casos reais, essas reduções alcançaram quase 50%. Espera-se que ferramentas CASE que façam uso do modelo helicoidal permitam economias ainda maiores. Entendemos, dessa forma, que o modelo de ciclo de vida helicoidal seja a mais poderosa e versátil ferramenta de gestão para projetos de sistemas de TI, em particular, e, possivelmente, para o desenvolvimento de qualquer sistema computacional, em geral.

Finalmente, pelos exemplos de redução a outros modelos de ciclo de vida que foram apresentados, provamos que o modelo de ciclo de vida helicoidal é o mais completo e geral desenvolvido até agora.

## Bibliografia

- [1] Ghezzi, C.; M. Jazayeri, M.; Mandrioli, D. *Fundamentals of Software Engineering* (1 ed, New Jersey, Prentice Hall, 1991).
- [2] Pressman, R. S. *Software Engineering* (New York, McGraw-Hill, 1992).
- [3] Peters, J. F.; Pedrycz, W.; *Software Engineering: An Engineering Approach* (1 ed. New York, John Wiley & Sons, 2000).
- [4] Boehm, B. W.; A spiral model of software development and enhancement, *IEEE Computer*, 21(5), 1988, 61-72.
- [5] Yourdon, E.; *Managing the system life cycle*. (2 ed. New York, Prentice-Hall, 1988).
- [6] Sommerville, I.; *Engenharia de Software*. (6 ed. São Paulo, Addison Wesley, 2003).
- [7] Dejesus, E. X.; Programação sem sustos, *BYTE Brasil*, 4(8), 1995, 130-136.

- [8] Alves, A.C.P.D.; A Low-Cost PWR Nuclear Power Plant Simulator for Initial Training of Operators and Educational Purposes, *Proceedings of XI Enfir*, Nova Friburgo, RJ, Brasil, 1997.
- [9] Alves, A.C.P.D.; Introducing the Helicoidal Life Cycle – A Tool for Management of Software Development, *Proceedings of 7th International Conference of Software Engineering and Applications – SEA 2003*, Marina del Rey, CA, USA, 2003.
- [10] Alves, A.C.P.D.; Managing Terms And Costs With The Helicoidal Life Cycle, *Proceedings of 26<sup>th</sup> Annual Conference of the The International Society of Parametric Analysts*, Frascati, Italy, 2004.