3
Conceitos e base teórica

Este capítulo estabelece os conceitos necessários para entender os elementos, os processos e a finalidade da metodologia da cadeia crítica CC/BM, tanto em projetos individuais como em ambientes de múltiplos projetos. Essa base conceitual é necessária para o acompanhamento dos capítulos seguintes

3.1.
A metodologia da cadeia crítica e gestão de buffers de tempo (CC/BM)

A metodologia da cadeia crítica surge como uma teoria alternativa ante a necessidade de aprimorar os resultados da abordagem clássica na gestão de projetos. Essa metodologia nasce da aplicação da "teoria das restrições" num sistema de planejamento e execução de projetos.

3.1.1.
Teoria das restrições - TOC (Theory of Constraints)

Segundo Leach (2005) a teoria é estruturada no entendimento de qualquer sistema produtivo. Estabelecendo, que todos os sistemas produtivos possuem, no mínimo, uma restrição que limita suas saídas ou *outputs*. Quer dizer, que se o sistema estivesse livre de restrições teria saídas infinitas ou nulas. A finalidade da TOC é aproveitar ao máximo a capacidade do sistema, fazendo que as saídas atinjam a meta estabelecida. A teoria aceita que qualquer otimização isolada das partes pode não gerar uma melhora nos resultados do sistema e que toda solução aplicada ao sistema tende a se deteriorar com o tempo em conseqüência das mudanças do ambiente. Essas premissas básicas fazem com que a teoria estimule uma melhora contínua dos processos e resultados do sistema através do ciclo persistente de procura e eliminação de novas restrições.

Com raiz nesses conceitos sistêmicos Goldratt cria o método de controle de produção chamado *Drum-Buffer-Rope*. Este método é composto pelo "tambor"

(*drum*) que representa a capacidade de processamento da restrição do sistema pela "corda" ou *rope*, que representa a informação do estado do tambor que libera o ingresso de trabalho no sistema com o objetivo de não deixar a restrição ociosa, finalmente, pelos *buffers*. Estes ao serem localizados em lugares estratégicos absorvem as flutuações não previstas no processo. (NEWBOLD, 1998; LEACH, 2005)

Leach (2005) explica como esse método se amolda as condições do mundo dos projetos, onde certamente existem restrições e flutuações estatísticas que evitam o término do projeto em menor tempo. Neste contexto a *restrição*, ou *tambor*, do sistema corresponde à cadeia crítica, os *buffers* são equivalentes a pulmões tempo e a "corda" ou *rope*, é a ordem de liberação de projetos baseada na capacidade disponível dos recursos, na execução de atividades em base ao estado dos *buffers* e na decisão de reprogramar a rede. Repare-se que para aplicar estes conceitos no mundo dos projetos é preciso aceitar que cada projeto é um sistema onde existem dependências entre seus elementos, e que toda mudança de atividades, recursos, políticas, etc. afeta o sistema.

A teoria define cinco passos importantes no processo de melhora contínua do sistema:

1 Identificação da restrição do sistema: Nos projetos a restrição do sistema comumente é formada pelos recursos e interdependências entre atividades que tem o impacto mais negativo em função de tempo, custo e escopo do projeto.

2 Exploração da restrição: O objetivo é desenvolver um programa (*schedule)* que permita obter o maior proveito da restrição mantendo-a o mais ocupada possível.

3 Subordinar tudo à restrição: Focalizar a atenção na restrição fazendo que todos os elementos do sistema trabalhem em função dela. Ou seja, programar todas as atividades e recursos não restritivos em função da cadeia ou seqüência crítica.

4 Elevar a restrição: Aumentar recursos e/ou resolver dependências das atividades.

5 Se a restrição é resolvida ir ao passo um.

3.1.2.
CC/BM em projetos individuais

Para poder entender os elementos técnicos e a estrutura de programação da metodologia é preciso entender primeiramente sua base teórica. Nesse sentido, Leach (2005), Herroelen *et al.* (2002) e outros, explicam a presença constante de atrasos e sobre-custos na execução de projetos por meio da combinação de três fenômenos: “A síndrome do estudante”, “A lei de Parkinson”, e a conhecida “Lei de Murphy”.

A **síndrome do estudante** se refere à tendência que as pessoas, especialmente as que estão constantemente ocupadas, têm para esperar que as atividades se tornem realmente urgentes antes de iniciá-las. Leach (2005) afirma que, geralmente, o trabalho realizado nos primeiros dois terços da duração total da atividade é menor a um terço do total do trabalho. Esta situação faz com que a probabilidade de reconhecer alguma variação positiva ou adiantamento na execução das durações das atividades seja extremamente pequena. Isso explicaria o motivo pelo qual raramente se observaram adiantamentos no termino das tarefas. A **lei de Parkinson** diz que o trabalho tende a se estender até cobrir (e muitas vezes exceder) todo o tempo permitido. Tal síndrome se vê reforçada por políticas comumente implantadas na gestão de projetos, onde o término antecipado de alguma tarefa é atribuído a uma estimativa excessivamente conservadora e significa um posterior aumento da carga de trabalho para o executor ou uma redução nas futuras estimativas de tempo. A **lei de Murphy** se refere à incerteza presente no projeto, esta assume que “*tudo que pode sair errado, certamente acontecerá*”. O verdadeiro sentido dessa lei é reforçar a necessidade de modelar um programa que proteja o cumprimento da meta estabelecida (em tempo, custo e escopo) das possíveis variações que possam acontecer durante a etapa da execução.

Segundo Herroelen *et al.* (2002) e Leach (2005), para evitar os efeitos não desejados resultantes da interação dos fenômenos mencionados anteriormente, é preciso observar três cuidados básicos dentro do programa:

- **Estimar as durações das atividades considerando apenas 50% de probabilidade do projeto não atrasar**.

  Tanto Leach (2005) quanto Herroelen & Leus (2001) e Herroelen *et al.* (2002) reconhecem que a duração das atividades tem um caráter estatístico com uma distribuição de probabilidade similar à da Figura 3.1. É assim que, tanto durações extremamente curtas, quanto durações extremamente longas têm uma chance apreciável de acontecer. Nesse contexto Leach (2005) adverte que as pessoas tendem a definir durações de atividades superestimadas com a finalidade de se proteger de futuras cobranças de desempenho. Isso ocasiona que a estimação da data de término do projeto seja exagerada, ademais, o fato de localizar essa proteção em cada atividade permite a ocorrência dos efeitos não desejados originados pela lei de Parkinson. A metodologia recomenda mitigar tais efeitos usando a média ou "tempo seco" na estimação das durações das atividades. Ou seja, subtrair o tempo de contingência considerado numa estimação protegida, a qual geralmente considera o 90% de confiança admitindo 10% de probabilidade de atraso. Tal contingência será posteriormente realocada num *buffer* específico que permita controlar melhor a variância inerente do projeto.

  Segundo Leach (2005) essa variância é originada por causas comuns ou naturais do sistema, sendo que sua gestão é o elemento essencial para o sucesso dos projetos. Por outro lado, Herroelen *et al.* (2002) questionam a utilidade de tal abordagem, alegando que durante a etapa de execução, certamente as distribuições das atividades não serão as mesmas.

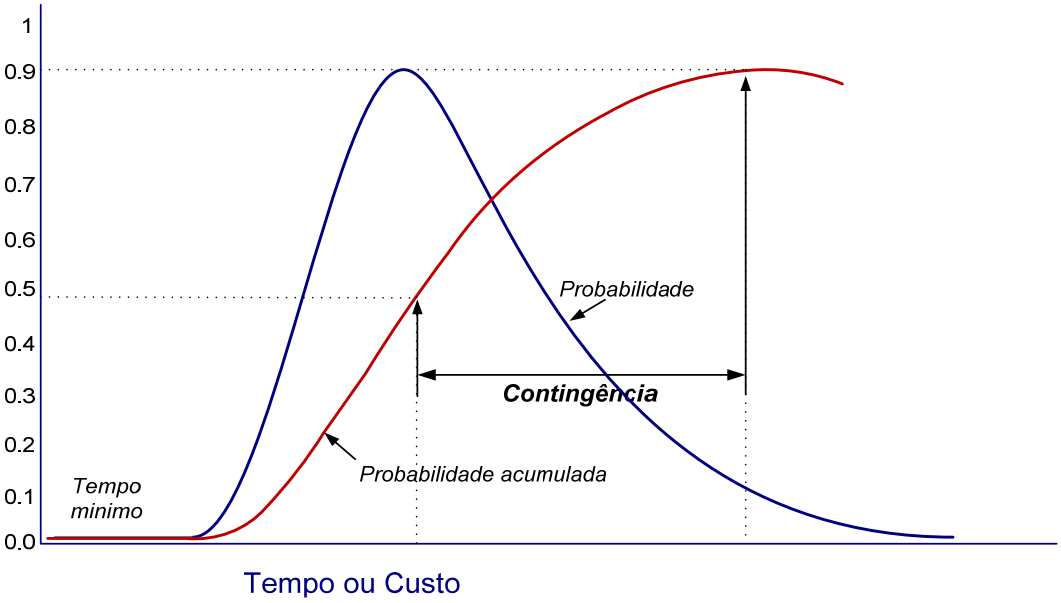

Figura 3.1: Distribuição de probabilidade das durações das atividades
Fonte: Leach (2005)

- **O programa não deve estabelecer datas para prometidas individuais, *due dates* ou *milestones*, que são atividades sem duração que marcam pontos de controle intermediários no projeto**.

  O programa da cadeia crítica estabelece datas apenas para o início das primeiras atividades de cada cadeia e para o fim do projeto (depois do buffer do projeto), para o resto das atividades se utilizam datas aproximadas que indiquem o início e o tempo faltante para completá-las. Isso se baseia na idéia de que o uso de *milestones* e *due dates* origina objetivos locais, fazendo com que os objetivos do sistema sejam relegados a segundo plano. Ademais, os controles intermediários evitam que ganhos de tempo obtidos por atividades terminadas antecipadamente sejam transmitidos ao longo das atividades sucessoras da cadeia, quer dizer, que mesmo tendo a atividade pronta para ser elaborada pelo recurso da seguinte atividade, este não estará preparado para trabalhar nela, pois a programação não contempla tal possibilidade. Conseqüentemente, o uso desses controles intermediários tende a facilitar o consumo do tempo disponível para executar uma atividade, tal como prevê a lei de Parkinson. (ANAVI-ISAKOW & GOLANY, 2003)

- **O programa não deve permitir a execução de múltiplas tarefas, ou seja, um único recurso não deve interromper uma atividade para fazer outra**.

  A origem desse comportamento é a pressão que os clientes e os encarregados dos projetos fazem sobre os recursos para priorizar seu projeto. Geralmente se pensa que trabalhar com múltiplas tarefas incrementa a eficiência e produtividade total. Leach (2005) entre outros mostram como dividir os esforços entre atividades, faz com que as atividades e os projetos sejam acabados mais tarde. Comparando a Figura 3.2 e Figura 3.3 se aprecia como os projetos, ao trabalhar-se simultaneamente em múltiplas tarefas, são estendidos consideravelmente. A duração dos projetos tende a aumentar quando se leva em conta os tempos de *set up* (tempo que demora o recurso para retomar a atividade interrompida).

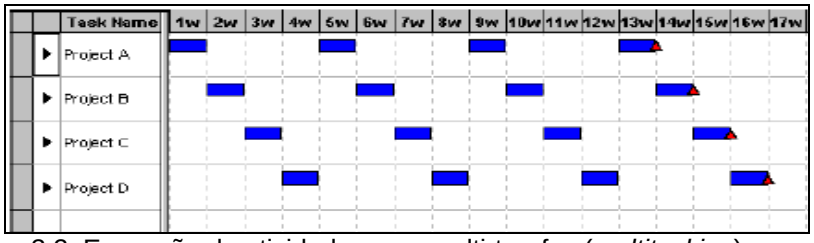

Figura 3.2: Execução de atividades com multi-tarefas (*multitasking*)
Fonte: http://www.sciforma.com

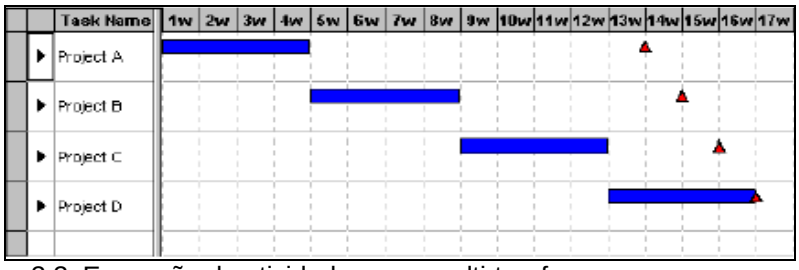

Figura 3.3: Execução de atividades sem multi-tarefas
Fonte: http://www.sciforma.com

3.1.2.1. Principais Elementos

Uma vez estabelecidas as condições básicas da metodologia, pode se definir os principais elementos que, junto ao processo de programação pretendem determinar uma data de termino de projeto que seja mínima e muito

provavelmente atingível. Para tal fim, a metodologia estabelece como premissa a necessidade de outorgar estabilidade ao programa mantendo níveis mínimos da carga de trabalho-em-processo (*work-in-process, WIP)*. (ANAVI-ISAKOW & GOLANY, 2003). A seguir alguns elementos são definidos de forma a tornar mais claro o resto deste texto.

- **Programação inicial de atividades**. - A metodologia recomenda programar as atividades usando as datas de termino mais tarde (*late-finish)* que permitam atingir a data de entrega prometida, levando em conta as restrições de precedência e de compartilhamento de recursos. Com base na programação "de trás para frente" se executam as atividades segundo o "sistema de produção puxado". É assim que se reduzo o nível de carga de trabalho-em-processo concentrando a maior quantidade de trabalhos simultâneos no final do programa, ou quando é estritamente necessário. Para reforçar tal prática, a metodologia estabelece que as atividades iniciais (todas aquelas que não têm predecessor) devem ser executadas exatamente na data estabelecida, e, as atividades intermédias, (que tem predecessores), devem ser executadas o mais cedo possível. Além disso, a maneira de controlar melhor a carga de trabalho, é necessário que a programação de atividades seja acompanhada pela resolução de conflitos entre os recursos, a qual também será feita "de trás para frente". Para lograr este nivelamento de recursos, Herroelen *et al.* (2002) mencionam a utilização de heurísticas, não bem definidas na literatura, que certamente implicam trasladar para a esquerda (mais cedo no tempo) as atividades que provocam o menor aumento da duração do projeto.

- **A cadeia crítica. -** A cadeia crítica é a seqüencia de atividades que, considerando as restrições de precedência tecnológicas e de dependência entre atividades pelo compartilhamento de recursos, determina a duração total do projeto. Essas atividades se caracterizam por não ter espaço suficiente para serem deslocadas no tempo, nem para o futuro (por ser programada com os tempos mais tarde) nem para o passado. Ou seja, são atividades sem folga que fazem parte da cadeia mais longa do programa.

De acordo com a “teoria das restrições”, é preciso subordinar o programa a um único elemento mais restrito, por conseguinte, a restrição do programa corresponde a uma única cadeia crítica. Caso exista mais de uma cadeia sem folga e com a mesma duração, será preciso escolher só uma delas. (NEWBOLD 1998; HERROELEN *et. al.*, 2002; ANAVI-ISAKOW & GOLANY, 2003; LEACH, 2005)

- ***Buffers*. –** Segundo Herroelen & Leus (2001) **s**ão pseudo-atividades que ao serem inseridas no programa fazem as funções de “estoques de tempo”. A inserção desse tempo extra no projeto procura assegurar a satisfação dos objetivos do sistema, evitando, tanto quanto, que as datas de entrega prometida sejam afetadas pelos possíveis distúrbios ou atrasos que possam acontecer durante a etapa de execução. Alem de proteger o programa, os *buffers* desempenham a função de controle durante o acompanhamento do projeto. Conforme se verá à continuação a alocação desta segurança em lugares estratégicos do programa permite obter informações oportunas sobre o estado do projeto e de suas partes críticas, facilitando o planejamento e aplicação de medidas corretivas quando o projeto estiver fora de controle. De acordo com a localização, definem-se três diferentes tipos de *buffers*: *Buffers* do projeto, *Buffers* de alimentação e *Buffers* do recurso.
  - O ***buffer do projeto – BP*** tem como função proteger o projeto de atrasos que possam acontecer na cadeia crítica. Para tal fim, eles são localizados logo depois da última atividade crítica, definindo assim, a data de entrega do projeto.
  - Os ***buffers de alimentação – BA*** têm a função de proteger a cadeia crítica de atrasos que possam acontecer nas atividades não críticas. Eles são localizados sempre onde as atividades não críticas desembocam na cadeia crítica.
  - Os ***buffers do recurso – BR*** se comportam de forma diferente dos anteriores, sua presença no programa não implica em um aumento de tempo de duração do projeto. Eles cumprem uma função meramente informativa. Sua localização é junto aos recursos que trabalham nas

atividades críticas, sempre e quando a atividade crítica anterior seja executada por outro recurso. A função dos BR é a de informar ao recurso o nível de avanço da atividade anterior, dando-lhe um horizonte de tempo para se preparar e dar início a sua atividade o mais rápido possível.

O dimensionamento dos *buffers* é calculado para refletir a incerteza presente na estimação das durações das atividades e do projeto, protegendo-o das variações comuns ou naturais do sistema. Existem dois métodos de dimensionamento formalmente apresentados na metodologia: "O método de cortar e colar" e "O método da raiz quadrada do erro". (TUKEL *et al*., 2006)

- O ***método de cortar e colar*** determina o tamanho dos *buffers* supondo que as estimações das atividades são feitas com uma probabilidade de atraso de 50%. O *buffer* é resultante da metade da soma de todas as contingências cortadas na estimação das durações das atividades que o alimentam. Kjersti & Taylor (1999), Herroelen *et al.* (2002), Raz *et al.* (2003) e Jyh-Bin (2007) advertem a tendência de crescimento linear que o *buffer* tem em relação ao comprimento da cadeia que o alimenta. Os autores indicam também como esse prazo excessivamente dilatado poderia significar a inserção de proteção desnecessária que reduza a competitividade dos projetos em comparação com empresas concorrentes.
- O ***método da raiz quadrada do erro*** usa duas estimativas da duração das atividades para determinar o tamanho dos *buffers*. A primeira é o "tempo seco" ou estimação da duração com 50% de probabilidade de atrasar ($\mathbf{A_i}$). A segunda estimativa inclui uma proteção suficiente para ser considera como duração de baixo risco (*low-risk*), esta é geralmente estimada com 90% de probabilidade de não atrasar ($\mathbf{S_i}$). A incerteza de cada atividade é calculada como a diferença de ambas estimativas ($\mathbf{S_i - A_i}$). Newbold (1998) postula que a duração das atividades tem uma distribuição lognormal e determina que a incerteza calculada equivale a dois desvios padrões. Com essa mesma lógica, o

autor estabelece que para cada atividade, o desvio padrão é aproximadamente igual à metade da incerteza calculada (**$(S_i - A_i)/2$**).

Para poder calcular o tamanho do buffer que possa cobrir a incerteza total da cadeia que o alimenta, o autor, fazendo uso do "teorema do limite central", postula que a soma das distribuições das atividades é representada por uma distribuição normal. Desta maneira, se estabelece que o tamanho do buffer equivale a dois desvios padrões, calculado da seguinte forma. Esse método de dimensionamento não pressupõe incertezas não correlacionadas.

$$\sigma = \sqrt{\left(\frac{S_1 - A_1}{2}\right)^2 + \left(\frac{S_2 - A_2}{2}\right)^2 + \ldots + \left(\frac{S_n - A_n}{2}\right)^2}$$

Existem outros métodos de dimensionamento de *buffers* propostos na literatura, mas, que ainda não têm sido aceitos formalmente pela metodologia. Tais propostas podem ser encontradas no artigo Tukel *et al.* (2006) onde os métodos de dimensionamento de *buffers* levam em conta as características de complexidade e do nível de compartilhamento de recursos dos projetos. Outras abordagens aplicam a teoria de filas e algumas ferramentas de simulação.

Alem da função de proteção, os *buffers* têm a finalidade de controlar o andamento do projeto, motivo pelo qual é preciso determinar pontos de gerenciamento específicos para o estado de consumo dos *buffer*s (Figura 3.4). Ao atingir o primeiro ponto de gerenciamento a metodologia recomenda prestar atenção no estado do programa e planejar ações futuras. Essas ações serão aplicadas, só, se o consumo do *buffer* atingir o segundo ponto de gerenciamento.

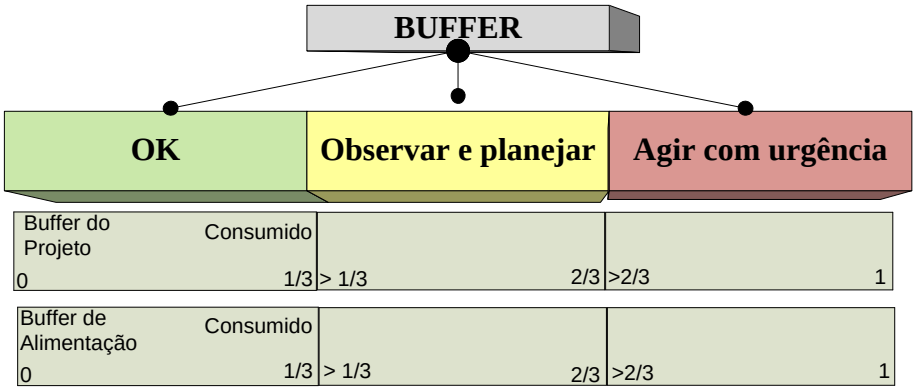

Figura 3.4: Exemplo clássico de gerenciamento de *buffers.*

Fonte: Barcaui & Quelhas (jul. 2004)

- **Mentalidade do papa-léguas (*Road-runner*).** - A idéia dessa mentalidade é permitir que o programa aproveite os ganhos de tempo conseguidos na cadeia crítica. Segundo Leach (2005), o autor da metodologia (Goldratt), define esse conceito como a regra de ouro da subordinação: *"Sempre que uma tarefa seja completada na cadeia crítica (e nas cadeias não críticas, sempre que a decisão não afete a cadeia crítica) as seguintes tarefas na seqüencia devem ser iniciadas o mais rápido possível"*. Quer dizer, que as atividades devem ser executadas o mais rápido possível, é claro, priorizando a cadeia crítica. Segundo Rand (2000). Herroelen *et al.* (2002) e Raz *et al.* (2003) a implementação dessa mentalidade implica que, além do programa normal (*schedule*) obtido com base nos tempos mais tardios, teria que se estruturar um segundo programa baseado nas atividades programadas com os tempos mais cedo possível (mentalidade do papa-léguas). Este programa, chamado como "programa projetado", reflete as atualizações feitas durante a etapa de execução.

3.1.2.2. O programa básico e o programa projetado

A combinação dos elementos acima descritos gera dois tipos de programas, que usadas simultaneamente, permitem focalizar, explodir e elevar a restrição do sistema fazendo com que os projetos sejam terminados dentro do tempo especificado (Figura 3.5). O **programa de base** ou programa básico é aquele que

estrutura a seqüência das atividades utilizando os tempos de início mais tarde e resolvendo os conflitos de compartilhamento de recursos. Nele também são inseridos os três tipos de *buffers* que determinam a duração total do projeto protegido contra as variâncias do sistema. O **programa projetado** é aquele que resulta da aplicação da mentalidade do papa-léguas, onde as atividades são programadas para serem iniciadas o mais rápido possível. Esse programa ao contrário do anterior, não leva em conta os *buffers* e é constantemente atualizado no acompanhamento da execução do projeto. Ambos os programas interagem entre si com a finalidade de controlar o andamento do projeto. O programa básico fica constante na sua estrutura, registrando as mudanças do programa projetado. Este informa a quantidade de trabalho executado, o tempo faltante para o término de cada atividade e o consumo dos *buffers*. Simultaneamente, o programa projetado recolhe essa informação e determina de forma dinâmica as atividades que podem ser executadas no futuro. Isso, subordinando a programação à cadeia crítica (restrição) do sistema.

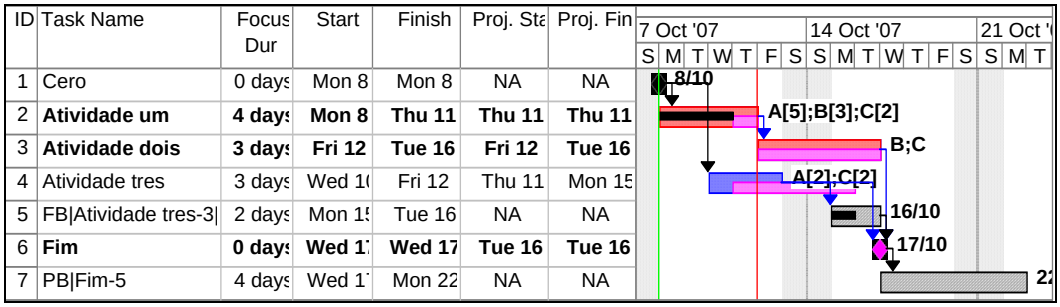

Figura 3.5: O *Programa base* e o *programa projetado*

As barras vermelhas (atividades críticas), barras azuis (atividades não críticas) e barras hachuradas (*buffers)* fazem parte do programa de base. As barras roxas fazem parte do programa projetado. As barras pretas indicam o trabalho executado nas atividades ou o consumo dos *buffers.*
Repare-se na primeira atividade como o programa projetado indica o trabalho a fazer segundo a informação do programa base. O consumo dos *buffers* é calculado em base ao programa projetado, na terceira atividade a barra roxa excede a linha vermelha indicando que essa atividade esta atrasada, o *buffer* é consumido na mesma quantidade.

3.1.2.3. Processo lógico de programação da cadeia crítica

A metodologia se inicia com a redefinição da duração de cada atividade, em forma determinística e sem folgas na estimação. A rede é construída levando em conta as restrições de precedência e de utilização de recursos. Os tempos de início

(*start time*) de cada atividade são programados usando a regra de “início na data mais tarde possível”. Com isso feito, identifica-se a **cadeia crítica** que é a seqüência de atividades que determina a duração do projeto. Nessa cadeia se insere o ***buffer* do projeto,** determinando sua data final. Posteriormente são inseridos os ***buffers* de alimentação**, essa inserção faz com que as cadeias não críticas que o alimentam, sejam empurradas no tempo (ou seja, para o lado esquerdo do programa). Muitas vezes a inserção dos BA faz com que o adiantamento do início das cadeias não críticas provoque novos conflitos de recursos. Em algumas ferramentas de *software* esta situação é definida como falta de espaço e é “solucionada” com a inserção de BA com certo grau de consumo. A segurança perdida nesta operação é recuperada no BP, quer dizer, que a quantidade de tempo que corresponderia aos BA é aumentada no BP. (HERROELEN & LEUS 2001; HERROELEN *et. al.* , 2002).

Finalmente, sempre que for necessário, incluem-se os ***buffers* de recursos** dentro da cadeia crítica. Algumas das ferramentas de *software* podem obter a informação oferecida pelos BR por meio de consultas predefinidas. Nesses casos pode se dispensar sua inserção, já que resulta uma complicação a mais na leitura dos diagramas de *Gantt* dos projetos. (Figura 3.6).

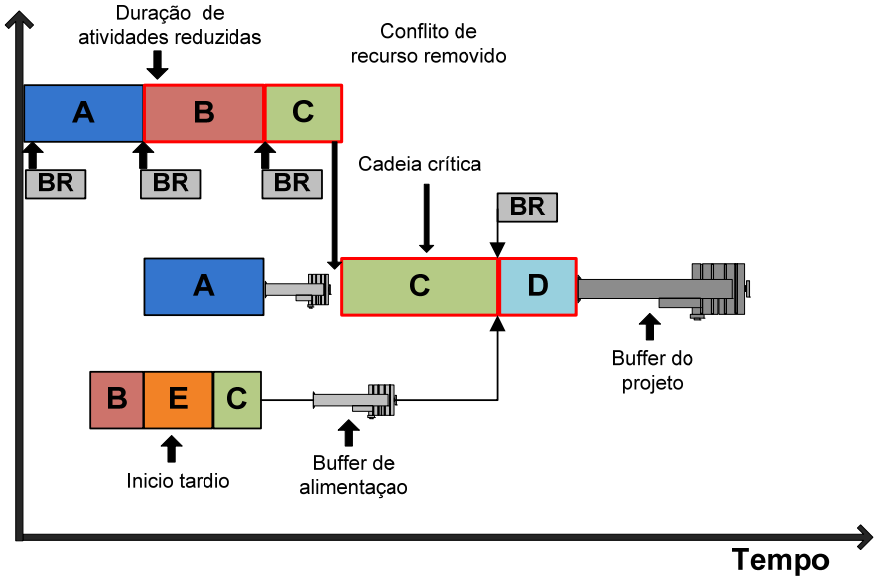

Figura 3.6: Principais elementos da metodologia CC/BM
Fonte: Leach (2005)
Os retângulos de diferentes cores indicam o tipo de recurso utilizado na execução das atividades, os retângulos com nomenclatura BR indicam os *buffer* de recurso utilizados. As atividades que formam parte da cadeia crítica são ressaltadas com um contorno vermelho e seguidas pelo BP. Existe uma restrição de capacidade no recurso C que faz com que a segunda atividade realizada por este seja deslocada. A ultima seqüência de atividades BEC mostram a subordinação da cadeia não critica à cadeia crítica, onde as atividades foram programadas com as datas de inicio tardio (*late start*) e se inseriu um *buffer* de alimentação

Uma vez definido o programa básico se inicia a execução do projeto. O acompanhamento da execução é realizado com a interação dos programas básicos e projetado. Segundo Newbold (1998) o critério de decisão na execução das atividades será priorizando as atividades críticas, em segundo lugar se executarão as atividades que provocariam um maior consumo dos *buffers* dos projetos e finalmente aquelas que provocariam um maior consumo dos *buffers* de alimentação.

3.1.3.
CC/BM em múltiplos projetos

A metodologia se estende também na programação de múltiplos projetos simultâneos que interdependem entre si pelo compartilhamento de um ou vários recursos. O sistema é composto por vários projetos que, do ponto de vista de restrições de seqüenciamento e de objetivos, são independentes entre si. O objetivo do sistema é maximizar o número de projetos executados segundo a

prioridade definida para cada projeto, mantendo o sistema em equilíbrio mediante o nivelamento da carga de trabalho entre os projetos. (LEACH, 2005).

Similarmente à programação de projetos individuais, a metodologia insere *buffers* de tempo localizados estrategicamente, procurando sincronizar a execução dos projetos, garantindo a continuidade do fluxo de projetos no sistema e o cumprimento das datas de término prometidas.

3.1.3.1. Principais Elementos

Ao se tratar de projetos simultâneos, alem dos elementos usuais da CC/BM alguns elementos adicionais são utilizados:

- **Projetos individuais corretamente programados**. - A programação dos múltiplos projetos parte dos projetos individuais corretamente programados de acordo com a metodologia CC/BM. Ao fazer esta primeira etapa, se considera que cada projeto individual conta com toda a disponibilidade de recursos do sistema. Desta maneira se assegura que cada projeto seja protegido contra as variâncias comuns ou naturais de suas próprias atividades.
- **A seqüência crítica**. - Uma vez que os projetos individuais são inseridos de forma prioritária (segundo o critério da empresa), pode-se reconhecer o recurso que representa a maior restrição para manter a continuidade do fluxo de projetos, isto é, o gargalo do sistema. Os proponentes da metodologia explicam que não se originam grandes complicações se esta eleição não é precisamente a correta. De qualquer maneira ao se aplicar os passos da “teoria das restrições” se revelará, eventualmente, o verdadeiro gargalo. Geralmente os critérios para a escolha do gargalo são baseados na demanda que o sistema tem pelo recurso, também pode ser baseada no custo financeiro que implica sua manutenção.

  Analogicamente à programação de projetos individuais, a metodologia procura reunir as restrições do sistema dentro de uma seqüência de atividades (como era na cadeia crítica). No caso de múltiplos projetos, esta é chamada de **seqüência crítica** e está formada por todas as

atividades de todos os projetos que são trabalhadas pelo recurso definido como gargalo.

- ***Buffer* de Capacidade – BC. -** Este buffer é definido em função do recurso gargalo (restrição do sistema) e tem a finalidade de evitar que, numa seqüência de atividades de diferentes projetos, um atraso numa atividade de um projeto venha atrasar outras atividades de outros projetos que a seguem. A meta do BC é inserir um grau de isolamento entre os projetos dos sistemas, fazendo com que a interdependência entre projetos, gerada pelo compartilhamento de recursos, seja menor. O BC permite também, manter a organização e equilíbrio dentro do sistema, evitando que a execução de atividades siga um comportamento reativo. Ou seja, indo de um projeto a outro tentando solucionar estados de emergência.

  A localização dos BC é entre dois projetos consecutivos, especificamente, entre a última atividade do primeiro projeto e a primeira atividade do segundo, sendo que ambas as atividades são executadas pelo mesmo recurso gargalo. Nas ferramentas de *software* não se utiliza uma representação gráfica para o BC, esse é simplesmente um espaço inserido entre as atividades dos projetos.

  O dimensionamento dos BC é baseado na capacidade do recurso gargalo, autores como Leach (2005) estabelecem como na prática é razoável que este seja determinado como 25% da capacidade do recurso. Outros autores como Newbold (1998) o determinam como uma porcentagem complementar à capacidade do recurso, quer dizer, se o desejado é reservar 25% da capacidade, se considerara que a carga atual do recurso corresponderá a 75% de capacidade. Sustentações matemáticas ou estatísticas que expliquem essas recomendações não foram encontradas na literatura.

- ***Buffer* Tambor – BT.-** Este *buffer* tem a função de proteger as atividades executadas pelo recurso estratégico de atrasos que possam acontecer nas atividades executadas por recursos não estratégicos. Além da função protetora, o BT assegura que o recurso estratégico se mantenha

> sempre ocupado, garantindo que seu fluxo de trabalho seja contínuo. A definição dos BT é similar à dos BA, estes ocupam um espaço na estrutura do programa e sua inserção implica empurrar para mais cedo no tempo (para a esquerda) a cadeia que o alimenta. Estudos sobre o comportamento deste tipo de *buffer* não foram achados na literatura pesquisada. Similarmente, na ferramenta de *software* utilizada, a inserção dos BT não é uma aplicação estritamente necessária como é o caso do seu análogo em projetos individuais. (*buffer* de alimentação – BA*).*

Na programação de múltiplos projetos persiste a utilização do "programa base" e do "programa projetado".

3.1.3.2. Processo lógico de programação da seqüência crítica (múltiplos projetos)

A metodologia se inicia com a programação de cada projeto individual segundo as recomendações da CC/BM. Os projetos são ordenados de acordo à prioridade outorgada. Essa prioridade é utilizada também como critério de decisão na escolha dos primeiros projetos a programar dentro do sistema de múltiplos projetos. Segundo Newbold (1998) tais prioridades devem se manter fixas durante todo o processo de desenvolvimento dos projetos, recomendando que sejam alteradas unicamente em casos extremos de perda de controle. Não obstante, autores como Cohen *et al.* (2004) consideram que a prioridade na escolha do próximo projeto (ou atividade) a programar seja feita com base do estado dos *buffers* do projeto- BP, priorizando aquela atividade que pertença ao projeto com o *buffer* mais comprometido.

Uma vez que algum tipo de critério é estabelecido na priorização dos projetos, eles são inseridos num programa especial que alberga o conjunto de múltiplos projetos. Diferentes abordagens, as que são explicadas no capítulo IV, são utilizadas na definição estrutural desse programa especial.

O seguinte passo na programação é a identificação do(s) recurso(s) que restringem a saídas do sistema (a definição da quantidade de restrições do sistema varia com a abordagem utilizada). Com base nessa restrição, no processo de nivelamento do sistema, serão determinadas as datas de início de cada projeto. A finalidade do nivelamento é equilibrar a carga de trabalho no sistema de tal

maneira que esta seja constante. Caso exista um conflito de recursos no sistema, os projetos de menor prioridade serão deslocados no tempo, atrasando sua data de início, mas, sem alterar sua programação interna. Uma vez que os projetos estão corretamente equilibrados, pode-se identificar a "**seqüência crítica"** de atividades (entre todos os projetos) que serão executadas pelo(s) recurso(s) gargalo.

Em base à seqüência crítica se determina o tamanho e localização dos *buffers de* tempo que irão a proteger o sistema. Primeiramente se inserem os *buffers* de capacidade - BC, ao aumentar o espaçamento entre os projetos se reduz a dependência, pelo compartilhamento de recursos, entre eles. Finalmente se inserem os *buffers* tambor – BT que similarmente aos BA provocam adiantamentos na data de início das cadeias que o alimentam. Da mesma maneira do que com os BA, existem casos em que não se conta com o espaço necessário para inserir BT do tamanho recomendado, infelizmente, não foram achados métodos alternativos para recuperar a proteção que seria perdida na inserção de *buffers* de menor dimensão. Herroelen *et al.* (2002), defendem que tanto para BA quanto para BT é necessário a aplicação de algoritmos especiais de reprogramação, não obstante, esse tema ainda não é muito estudado na literatura.

O acompanhamento do programa de múltiplos projetos também é realizado com a interação do programas básico com o programa projetado de cada projeto. A ferramenta de *software* utilizada atualiza o programa projetado de acordo com as prioridades (fixas) definidas inicialmente para cada projeto. Cohen *et al.* (2004) atualizam o programa projetado (no seu experimento) considerando prioridades dinâmicas em base à "seqüência crítica" e às atividades dos projetos de maior consumo de BP e BA.

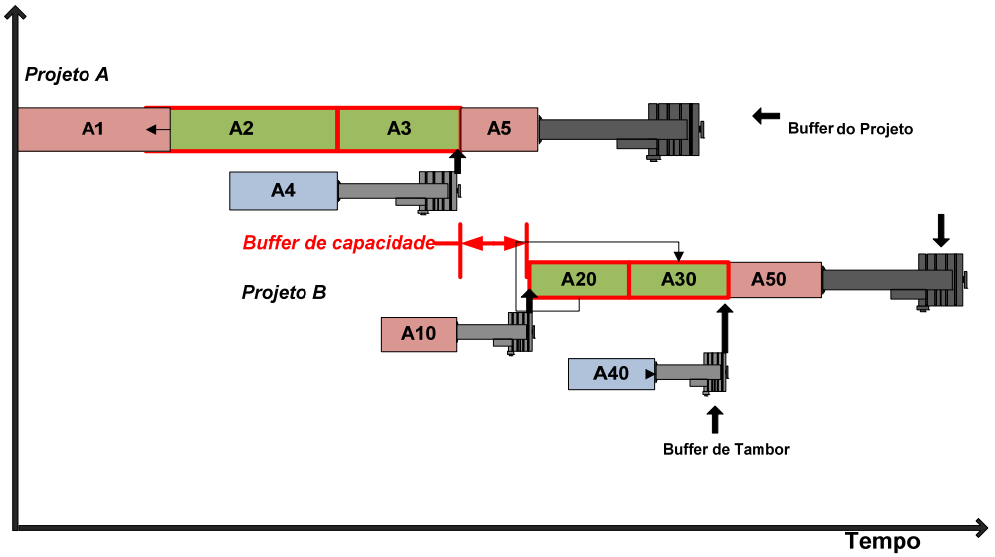

Figura 3.7: Principais elementos da metodologia CC/BM em múltiplos projetos.
Fonte: Leach (2005)

Três diferentes recursos são utilizados na execução de dois projetos simultâneos, estes são diferenciados pela cor do retângulo (rosa, verde e azul), sendo que o recurso verde é definido como “gargalo” do sistema e suas atividades formam a “seqüência crítica”. Ademais dos BP de cada projeto, insere-se o BC entre as atividades executadas pelo mesmo recurso, mas de diferentes projetos, esse é representado por um espaço delimitado por linhas vermelhas. Os BT são inseridos nas cadeias que desembocam na seqüência crítica, empurrando-las à esquerda.