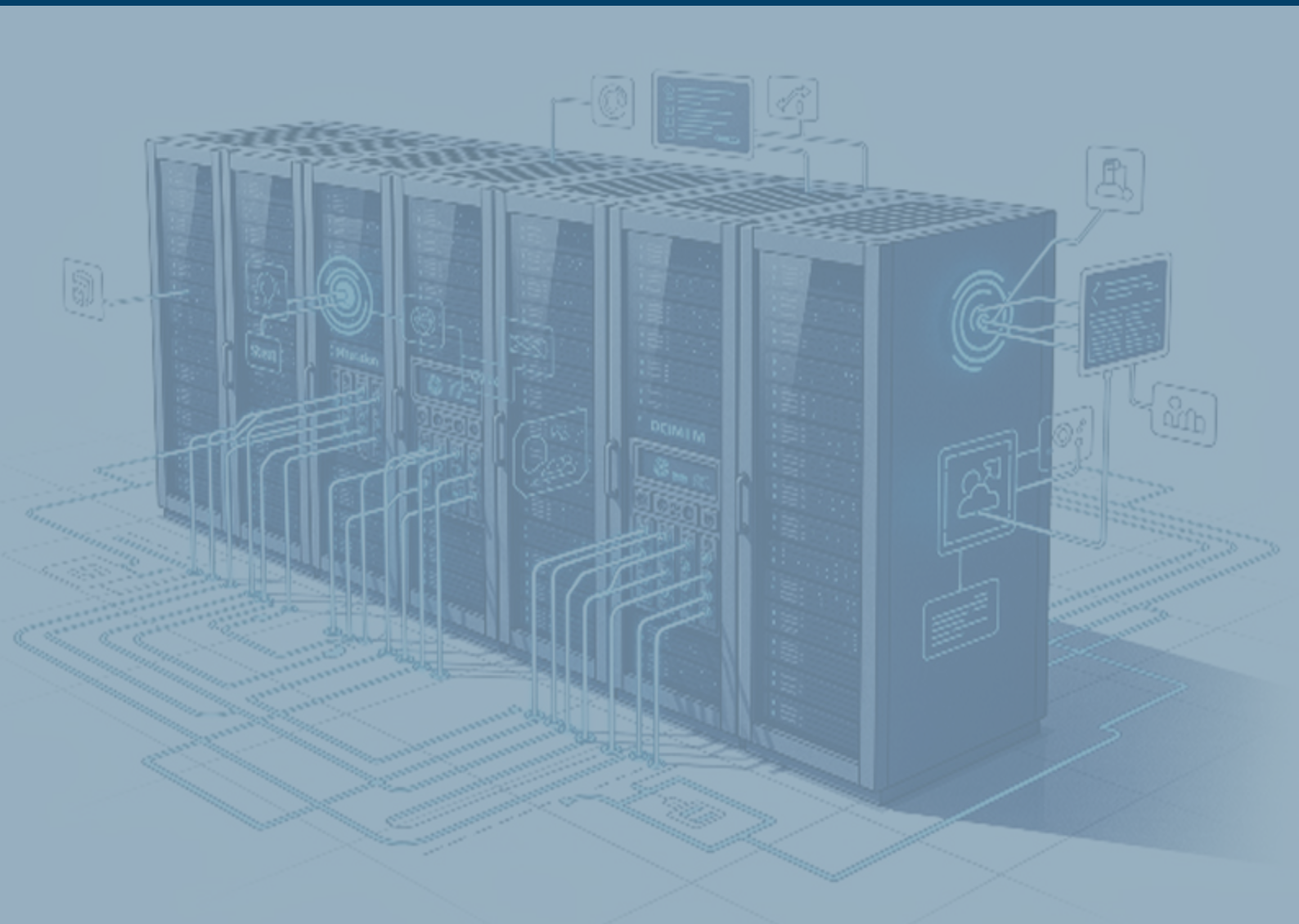
Bjumper

Riscos na implementação de um DCiM e estratégias para mitigá-los

Modelar, monitorar e integrar com critério



Introdução

Implementar um DCiM não é apenas um projeto tecnológico, é uma mudança na forma de entender e gerir a infraestrutura, e como em qualquer mudança relevante surgem dúvidas, resistências e riscos.

É normal, o importante não é negar que eles existem, mas saber identificá-los a tempo e abordá-los com método e bom senso.

Com uma abordagem prática e bem estruturada, esses riscos deixam de ser uma ameaça e passam a se tornar parte natural do processo de maturidade operacional.

Um DCiM agrega valor quando consegue três coisas ao mesmo tempo:

1. **Descreve** bem a sua realidade (modelar)
2. **Mede** bem o que acontece (monitorar)
3. **Conecta** esse dado com o restante da organização (Integraciones)

Não é “instalar uma ferramenta”, **é transformar conhecimento disperso em operação controlada.**

Camadas do projeto

Uma forma de controlar os riscos de um projeto de DCiM é considerar as diferentes camadas ou fases, que não precisam ser consecutivas, mas podem ser realizadas em paralelo, sempre com acompanhamento paralelo.

Vamos ver as primeiras, embora isso seja apenas uma questão de critério.

Modelar

Construir o “mapa” do Data Center:

A primeira pergunta que se deve fazer é**: tenho a informação que o modelamento exige? Onde ela está? Quem a possui? Está atualizada?**

Informação necessária:

- Estrutura: site / sala / fila / rack.
- Cadeia elétrica (UPS, quadros, PDUs, racks)
- Definição clara de relações (o que depende de quê).
- Inventário TI (e/ou infra conforme o escopo).
- Padrões de nomenclatura e classificação.
- Conectividade básica.

Isso significa que você precisa ter tudo antes de iniciar um projeto de DCiM? Não, grande parte será organizada e levantada durante o desenvolvimento do DCiM. O que isso significa é que você deve estar consciente, antes de começar, do que tem e do que está faltando.

Risco habitual: Modelar de forma incompleta ou desatualizada transforma o DCiM em uma “foto bonita”, mas inútil para a operação. Não inclua dados nos quais você não tenha confiabilidade.

Entrega (marco de qualidade): Modelo coerente vs realidade

• Layout: Tamanhos e distribuição das salas, áreas ocupadas pelos equipamentos de facilities, racks, distribuição elétrica (se for aérea) e distribuição de ar (placas perfuradas/dutos elevados/InRows/corredores confinados).

• Rastreabilidade elétrica: Como a energia chega desde a entrada (fornecedor/gerador) até o elo final (rPDUs/bornes/ficha steck/equipamentos de AA), considerando redundâncias.

• Rastreabilidade de rede: Como a conectividade chega desde a entrada (Carriers/escritórios) até o elo final (computação/storage).

• Padrão de naming IT: DC/sala/fila/rack/posição U.

• Dados mínimos completos dos ativos: Localização, Tipo, marca, modelo, IP, etiquetas.

Monitorar

Coletar dados diretamente do hardware por protocolos padrão (leituras ao vivo e confiáveis):

Informação necessária:

- Captura confiável de consumos elétricos, temperatura, carga, estado dos equipamentos..
- Limiares coerentes com a realidade operacional.
- Alarmes acionáveis (não ruído constante).
- Correlação entre eventos físicos e lógicos.

Risco habitual: Excesso de dados sem análise equivale à saturação operacional e perda de confiança na ferramenta.

Entrega (marco de qualidade)

• Sinais corretos (valor/unidade/escala).

• Estabilidade de comunicação: Configuração correta dos parâmetros de rede e equipamentos. Tempo entre polling de dados compatível com a quantidade de dispositivos.

• Alarmes com sentido: Configuração correta das políticas de notificação conforme criticidade. Evitar telas ou caixas de e-mail sobrecarregadas com alarmes não críticos (limiares, desconexões programadas, entrada em manutenção, etc).

• Dados mínimos completos dos ativos: Localização, Tipo, marca, modelo, IP, etiquetas.

Integrações

Um DCiM isolado perde grande parte do seu potencial, o verdadeiro valor aparece quando o dado flui para outros sistemas. Aqui é onde o DCiM começa a se tornar o cérebro do Data Center.

Integração BMS com DCiM (unidirecional)

O que o BMS aporta

dados de facilities (clima, energia/estado em certos elementos, alarmes do ambiente, etc. conforme cada BMS).

O que o DCiM aporta

Contexto operacional:

- “Onde ocorre?”
- “Quais racks/equipamentos dependem?”
- “Qual serviço pode ser afetado?”
- “Qual capacidade/energia está comprometida?”

Risco habitual

ntegração “funciona” mas não é útil (eventos sem contexto ou métricas mal interpretadas).

Boas práticas

- definir quais sinais “vão para operação” (os que realmente são monitorados).
- normalizar unidades e limiares desde o início.
- piloto: 1 sala / 1 conjunto de pontos ---- validar --- escalar.

Integração CMDB com DCiM (bidirecional)

O que aporta a CMDB (lógico)

- serviço, aplicação, owner, criticidade, estados IT
- relações lógicas (CI a serviço, dependências).

O que aporta o DCiM (físico)

- ocalização (site/sala/rack/U).
- dependências elétricas, portas físicas / patching (se aplicável).
- contexto físico (ambiente, energia, capacidade).

Valor real

fechar o gap entre “o que a TI acredita que tem” e “o que fisicamente existe e suporta o serviço”.

Riscos típicos

- conflito de “source of truth” (quem manda em qual campo?).
- dados duplicados ou sobrepostos (naming, estados, owners).
- bidirecional sem regras --- transforma-se em “ping-pong de dados”.

Boas práticas

- Definir ownership por campo (isso evita 70% dos problemas).
  Exemplo típico:
  Localização física ---- DCiM manda.
  Owner / serviço / criticidade --- CMDB manda.
- Definir o “ID comum” (serial, asset tag, CI ID…) para reconciliação.
- Definir eventos de sincronização (alta, baixa, mudança, movimentação).
- Critério de aceitação: “os ativos-chave coincidem e os campos não se sobrepõem”.
- Projetar integrações baseadas em casos de uso reais, não em “possibilidades técnicas”.

Quando integrado corretamente, o DCiM deixa de ser uma ferramenta de infraestrutura e passa a ser um sistema de suporte à tomada de decisão.

Os blocos que mais se repetem

Pré-requisitos e acessos

Antes de começar com o DCiM, precisamos que a base esteja preparada e é aqui que mais bloqueios aparecem.

- Arquitetura e plataforma prontas (servidores provisionados, versões validadas, recursos corretamente dimensionados).
- Licenças disponíveis e corretamente atribuídas.
- Acesso remoto operacional (VPN, saltos, credenciais).
- Comunicações habilitadas entre as redes envolvidas (IT, OT, gestão).stión).

Muitas vezes não é um problema técnico complexo, mas pequenas dependências que ninguém fechou formalmente.

Regra:
Esta fase pode ser bastante frustrante. Uma forma muito útil de desbloquear é identificar rapidamente as pessoas adequadas dentro da sua organização e colocá-las em contato direto com as pessoas do fornecedor que precisam dessa validação. Na Bjumper, pelo menos, trabalhamos assim: menos e-mails cruzados e mais conversa direta e, se for presencial, melhor ainda.

Coleta de dados para modelar

O modelamento depende diretamente da qualidade do dado que recebemos.

- Inventário e layout atualizados + sessões de esclarecimento de dúvidas.
- Conectividade de rede documentada (o que conecta com o quê) + revisão conjunta para resolver incoerências.
- Documentação elétrica (diagramas unifilares atualizados).
- Plantas CAD e qualquer detalhe físico necessário para representar corretamente sala, racks e distribuição.

Na prática, quase sempre aparecem diferenças entre o documentado e a realidade, e é normal, o importante é detectá-las antes de carregar o modelo.

Regra:
O DCiM não corrige dados ruins, ele os amplifica. Se o inventário está desalinhado, o erro não desaparece, pelo contrário, torna-se mais visível.

Monitorar: medir bem antes de medir tudo

Um dos erros mais comuns é querer integrar todos os sinais desde o primeiro dia. Para que a monitorização funcione:

- Os protocolos devem ser padrão e estar bem configurados (por exemplo, SNMP corretamente habilitado).
- As tabelas SNMP ou MIBs devem estar identificadas, acessíveis e documentadas.
- As credenciais e comunidades devem estar validadas.
- Os sinais críticos devem estar testados antes de escalar.
- Decidir quais dados realmente agregam valor ao DCiM. Nem tudo o que pode ser medido deve ser integrado.

Por exemplo:

Faz sentido integrar consumo por PDU, carga por ramal e temperatura por corredor se queremos fazer capacity planning energético.
Não faz sentido começar integrando 200 sensores secundários se ainda não temos claro qual KPI queremos explorar.

Regra:
Primeiro “poucos sinais, bem feitos”. Depois escalamos.

Integrações: o truque é delimitar

Integrar com o BMS, com a CMDB ou com qualquer outro sistema não é um marco em si. A integração técnica não é o objetivo. O verdadeiro marco é ter um caso de uso operacional funcionando, com critério de aceitação claro.

Por exemplo:

- Que uma mudança na CMDB atualize automaticamente o modelo físico validado.
- Que um alarme crítico do BMS gere um evento correlacionado e acionável no DCiM.
- Que uma ampliação de carga impacte automaticamente em um relatório de capacidade.

Se não houver um caso de uso concreto por trás, a integração se transforma em um “check” de projeto sem impacto real.

Regra:
Não integrar por integrar. Integrar para resolver algo concreto.

Como transformar riscos em marcos acionáveis

Um risco em um projeto DCiM não desaparece por ser identificado em um documento. Ele é reduzido quando o transformamos em algo concreto, mensurável e com responsável..

A diferença entre um projeto que avança e um que se bloqueia geralmente está aqui, em como passamos de “isso pode ser um problema” para “isso tem dono e data”. Para cada marco, especialmente em dados, monitorização e integrações, o mínimo que deve existir é:

Dono:

Uma pessoa específica, não uma área, não “a equipe de IT”, não “Facilities”. É necessário nome e sobrenome porque, se algo não tem dono, na prática não existe e, se tem vários donos, provavelmente ninguém irá priorizá-lo.

Data:

Data realista e acordada, não imposta. É melhor uma data consensual do que uma data otimista que seja descumprida três vezes. Além disso, convém diferenciar entre:

- Data de entrega técnica.
- Data de validação funcional.

Porque nem sempre coincidem.

Dependências:

Aqui é onde mais surpresas aparecem, porque um marco de integração pode depender de:

- Abertura de firewall.
- Entrega de credenciais.
- Ativação de SNMP.
- Validação de um inventário prévio.
- Disponibilidade de um ambiente intermediário.

Se as dependências não estiverem explicitadas, o atraso parece “inexplicável”, quando na realidade estava anunciado desde o início. Colocá-las por escrito ajuda todos a entenderem o que precisa acontecer antes de dar o próximo passo.

Critério de aceitação:

Este é o ponto mais importante e o menos definido, pois não basta dizer “integração concluída”. A pergunta é: o que exatamente precisa acontecer para considerar que está correto?

Exemplos:

- Os dados são recebidos?
- São recebidos com a periodicidade correta?
- Estão mapeados corretamente no modelo?
- Geram alarmes coerentes?
- O usuário final consegue utilizá-los?

Se não definirmos isso antes, a validação se transforma em uma discussão subjetiva.

Delivered ≠ Validated

Que algo esteja entregue não significa que esteja validado.

- Pode estar instalado, mas não testado sob carga real.
- Pode estar integrado, mas sem dados consistentes.
- Pode estar modelado, mas sem revisão por parte do responsável da sala.

A validação implica que:

- O responsável o revisou.
- Foi testado em condições reais.
- Cumpre o critério de aceitação acordado.

Até que isso aconteça, o risco não está encerrado. Apenas foi “movido”.

Transformar riscos em marcos acionáveis não é burocracia de projeto, é a forma mais prática de evitar bloqueios longos e silenciosos. Quando há dono, data, dependências claras e critério de aceitação definido, o projeto ganha tração e o DCiM deixa de ser uma iniciativa ambiciosa para se tornar uma implantação controlada.

Checklist

Este checklist não é teórico, é o mínimo que deveria estar fechado antes de dar cada bloco como “completo”. Se faltar algum desses pontos, o normal é que o problema apareça mais adiante.

Modelar

Padrão de naming e hierarquia

Definir como são nomeadas salas, racks, PDUs, equipamentos e circuitos antes de começar a carregar informações.
Deve existir uma hierarquia clara (site > sala > fila > rack > equipamento, por exemplo) e regras homogêneas.
Se cada área nomeia de forma diferente, o modelo acaba sendo inconsistente e difícil de manter.

Dataset de inventário/layout completo

Não apenas um Excel com equipamentos, deve incluir:

- Localização física exata.
- Consumo nominal (se aplicável).
- Conectividade básica.
- Estado operacional.

Quanto mais estruturado vier o dataset, menos retrabalho haverá depois.

Documentação elétrica mínima (conforme escopo)

Nem sempre é necessário o nível máximo de detalhe, mas pelo menos:

- Diagramas unifilares atualizados.
- Identificação clara de linhas, proteções e redundâncias.
- Relação entre rack e alimentação real.

Se o escopo incluir capacidade energética, este ponto deixa de ser opcional..

Sessão de validação conjunta do modelo

Antes de considerar o modelamento encerrado, deve haver uma sessão com quem realmente conhece a sala. Não basta que o modelo “bata na ferramenta”. Ele precisa estar alinhado com a realidade operacional, pois uma revisão conjunta evita muitas correções posteriores.

Monitorar

Lista de fontes HW e sinais prioritários

Não se trata de integrar tudo desde o início, deve existir uma lista clara de:

- Quais dispositivos são integrados primeiro (PDUs, UPS, climatização...).
- Quais sinais são críticos (consumo, carga, temperatura, estado).
- Qué señales son secundarias y pueden esperar.

Isso evita saturar o sistema e a equipe.

Conectividade / rede / permissões

Muitas integrações falham não por configuração, mas por permissões incompletas. Por essa razão aqui são validados:

- Acessos SNMP ou protocolos definidos.
- Aberturas de firewall necessárias.
- Credenciais e permissões confirmadas.
- Latência e estabilidade de rede.

Piloto validado (unidade / escala / estabilidade)

Antes de escalar, deve existir pelo menos um piloto funcionando corretamente:

- Dados coerentes nas unidades (kW, A, °C...).
- Periodicidade correta.
- Sem perdas intermitentes.

Se o piloto não for estável, escalar apenas multiplica o problema.

Plano de escalonamento

Definir por fases:

1. O que será integrado após o piloto.
2. Em que ordem.
3. Com qual critério de validação.

Escalar sem plano costuma gerar ruído e sobrecarga operacional.

Integrações

Aqui é onde mais se tende a simplificar e onde mais convém concretizar.

BMS com DCiM (unidirecional)

Deve-se definir:

- Sinais realmente prioritários.
- Unidades corretas e coerentes.
- Limiar alinhado com a operação real (não valores genéricos).
- Piloto prévio antes de integração massiva.

Integrar 500 sinais mal parametrizados não agrega valor.
Integrar 20 bem definidos, sim.

CMDB con DCiM (bidirecional)

Aqui o controle deve ser ainda maior.
Deve existir:

- Ownership por campo (quem é mestre de qual dado).
- Identificador comum único (ID compartilhado e estável).
- Regras claras de sincronização (unidirecional ou bidirecional).
- Teste de não colisão antes de passar para produção.

Se não for definido quem manda sobre cada campo, a sincronização gera conflitos silenciosos que são detectados meses depois.

Este checklist não busca burocracia, busca evitar retrabalho, discussões posteriores e “achávamos que isso já estava fechado”. Quando esses pontos estão claros, o projeto flui. Quando não estão, o DCiM deixa de ser uma solução e se torna uma nova fonte de incidentes.

O objetivo deste documento é revisar a ideia de que um projeto DCiM é complexo em seu desenvolvimento. Se esses passos forem seguidos e houver colaboração entre quem implementa e o usuário do sistema, será alcançada uma implantação fluida cujo resultado será ter uma tecnologia que suporte seus processos e facilite a operação para as pessoas.

Assim como dará visibilidade e controle do estado do Data Center no presente e no futuro. O que será alcançado será:

1. Minimizar riscos.
2. Reduzir custos.
3. Aumentar e melhorar o nível de serviço do Data Center.