O artigo Big Ball of Mud disserta sobre a construção por partes de software e como o fato dela ser incremental pode levar a um acúmulo gradual de problemas em sua arquitetua e eventualmente a uma grande desorganização do código (Big Ball of Mud). Ele aponta que a resolução desse problema está diretamente relacionada a um compromisso contínuo dos desenvolvedores com a consolidação e a refatoração do software. Existem vários padrões que podem surgir da desorganização (Big Ball of Mud), por exemplo o "Son of Big Ball of Mud" que surge quando uma equipe tenta construir um novo sistema a partir do código-fonte de um sistema antigo sem arquitetura definida; o "Linguistic Mismatch" que ocorre quando diferentes partes do sistema usam diferentes termos e conceitos para se referir aos mesmos objetos; e o "KEEP IT WORKING", que é a prática de fazer pequenos ajustes em um sistema para mantê-lo funcionando sem considerar a sua arquitetura geral. De modo geral, esses padrões emergem como tentativas de trabalhar dentro das limitações ou problemas da Big Ball of Mud e podem servir como sinais de alerta indicando a necessidade de reforma arquitetural. O foco principal desse artigo rodeia sempre a Big Ball of Mud e seus impactos de utilizá-la como padrão em sistemas de software, ele explicita muito sobre as forças que levam a emergência desse padrão e ao seu uso efetivo. Há menção de outras abordagens arquiteturais no artigo, mas seu objetivo é menos discutir essas abordagens em profundidade e mais entender por que tantos sistemas de software acabam sem arquitetura distinta. Além disso, o autor faz uma reflexão sobre possíveis alternativas para resolver este problema que não incluem o uso de abordagens arquiteturais hierárquicas, que nem sempre são necessárias ou viáveis na prática. Além disso em uma parte eles fornecem informações sobre o contexto para a discussão da Big Ball of Mud e das alternativas às abordagens arquiteturais tradicionais. Uma parte que me chamou atenção foi a discussão sobre a prática do desenvolvimento de throwaway code, que seria a produção de código descartável, muitas vezes praticada como solução temporária para resolver um problema específico ou como protótipo de um sistema. O arquivo aponta que essa prática é comum e pode ser útil em certas situações, mas pode levar a problemas sérios de manutenção no futuro se o código for promovido a uma solução de longo prazo. O texto também menciona que o código descartável pode ser uma espécie de "Kleenex Code", um código supérfluo e inútil que é difícil de remover depois de criado, e que as equipes de desenvolvimento devem ser conscientes dos custos e benefícios dessa abordagem. Outro termo citado é o da técnica de "Piecemeal Growth" que é uma abordagem de desenvolvimento de software incremental e iterativa que visa atender às necessidades de mudança do sistema à medida que ele evolui com o tempo. A técnica enfatiza a adição gradual de novas funcionalidades, em vez de uma grande reformulação, como forma de manter a flexibilidade e a fluidez do sistema. No entanto, o arquivo alerta para os riscos associados a essa abordagem, já que a adição gradual de novas funcionalidades pode, com o tempo, levar a um declínio gradual na estrutura geral do sistema e a uma perda de coesão arquitetural. Para mitigar esse risco, o arquivo enfatiza a importância de uma abordagem de refatoração contínua e consolidação para reverter a erosão arquitetural e manter a integridade do sistema. Como abordei anteriormente, o "Keep it Working" é a importância de manter sistemas de software funcionando em produção, em vez de realizar grandes reformulações ou reescritas. Essa abordagem enfatiza a manutenção do software existente, enquanto se concentra na solução gradual de problemas e na melhoria gradual das

funcionalidades. O arquivo argumenta que essa abordagem é muitas vezes mais prática e eficaz do que tentar substituir todo o sistema de uma vez, especialmente quando o software é complexo e tem muitas dependências no negócio. Uma prática citada que temos contato diretamente no ecossistema de desenvolvimento de software é o "sweeping it under the rug" que é uma metáfora para a tentativa de esconder ou suprimir problemas de software, em vez de resolvê-los adequadamente. O texto menciona que essa prática pode ser motivada por vários fatores, como prazos apertados, falta de habilidade técnica, preguiça ou falta de vontade de resolver o problema de forma adequada. Concluindo, o artigo destaca a complexidade da evolução de um software bem como a importância de abordar essa complexidade de forma adaptativa. Destaco a importância de gerenciamento de risco e de um equilíbrio entre os objetivos técnicos e os objetivos de negócio. Por fim, o texto enfatiza a importância da colaboração, da comunicação e do envolvimento de todas as partes interessadas do projeto para o sucesso do desenvolvimento de software.