CITIUS Plus

Análise à Auditoria e Proposta Técnica de Desenvolvimento/Migração do Projecto H@bilus/Citius para o Projecto Citius Plus e Análise do Código Fonte do Citius Plus

Intervenientes

Versão	Nome	Descrição
1.0	António Sardinha	Relator e Verificação Técnica
1.0	Jorge Afonso	Relator
1.0	Marco Figueiredo	Verificação Técnica
1.0	Mário Ferreira	Análise Técnica ao Código Fonte
1.0	Fábio Cardoso	Análise Técnica ao Código Fonte
1.0	João Rodrigues	Análise Técnica ao Código Fonte

Índice Analítico

1. Introdução

Um novo sistema é desenvolvido para solucionar um problema ou um conjunto de problemas que a organização reconhece num determinado momento. Para tal, é necessário definir e compreender o problema, desenvolver soluções alternativas, escolher a melhor solução e implementar a abordagem escolhida.

A equipa que ao longo de dez anos desenvolveu o projecto Habilus/CITIUS, colaborou com a CSW, de forma abnegada e empenhada, com o objectivo que sempre norteou a sua actividade: contribuir para o bom funcionamento do sistema judiciário, para a sua modernização e para a satisfação de todos aqueles que de forma directa ou indirecta usufruem do mesmo.

A participação desta equipa não pressupunha e não condicionou a abordagem escolhida. Limitou-se, de forma empenhada e colaborativa, a fazer a passagem do conhecimento que lhe foi solicitado. Não teve qualquer intervenção, nem sequer opinião, ao nível dos desenvolvimentos relacionados com o código fonte, do qual tomou conhecimento quase um ano depois de os trabalhos se terem iniciado.

A experiência consubstanciou uma oportunidade enriquecedora de troca de conhecimento, de contacto com outros métodos de trabalho e um estimulo adicional para, com a melhor receptividade, obter alguma valorização profissional, ainda que essencialmente conseguida através do esforço individual e colectivo, com muita autoformação e troca de opiniões.

Numa atitude construtiva e cooperante, no sentido de apresentar a sua visão sobre o resultado final e sobre a colaboração solicitada, sustentada pela sua experiência na área e pela análise crítica do trabalho até agora efectuado, vem assim, de forma que se pretende isenta e meramente técnica (sem melindre para a equipa da CSW, cujo mérito desde já reconhece), comentar alguns aspectos que, a seu ver, não coincidem com as boas práticas geralmente aceites e com as expectativas geradas na proposta apresentada pela Critical Software e contratada pelo Ministério da Justiça.

1.1 Objectivo

Dotar o ITIJ, IP de informação para suporte à decisão, documentando alguns factos e fazendo uma breve avaliação do produto apresentado até esta fase (piloto, pré-aceitação).

1.2 Referências

Data	Tipo	Descrição	Autor
N/A	Documento	Application Architecture Guide, 2nd Edition.pdf	Microsoft
N/A	Web Site	http://msdn.microsoft.com/en-us/library/microsoft.visualbasic.compatibility.vb6.support.aspx	Microsoft
N/A	Web Site	http://msdn.microsoft.com/en-us/library/ee839621.aspx	Microsoft
N/A	Web Site	http://pt.wikipedia.org/wiki/Orienta%C3%A7%C3%A3o_a_objetos	Wikipedia
N/A	Web Site	http://support.microsoft.com/kb/972959/pt	Microsoft
N/A	Web Site	http://msdn.microsoft.com/en-us/library/ff921345.aspx (patterns & practices)	Microsoft
N/A	Web Site	http://msdn.microsoft.com/en-us/library/8fxztkff(v=vs.71).aspx	Microsoft
N/A	Web Site	http://msdn.microsoft.com/en-us/magazine/cc188717.aspx	Microsoft
N/A	Web Site	http://msdn.microsoft.com/en-us/library/ms973912.aspx	Microsfot

2. Análise à Auditoria e à Proposta Técnica da Critical Software (CSW)

A equipa de desenvolvimento da DGAJ, agora do ITIJ, apenas conheceu a Auditoria (cujo último aditamento é datado de 04 de Março de 2010) e a Proposta Técnica de Migração Habilus/Citius (datada de 26 de Fevereiro de 2010), em 23 de Julho de 2010, durante o arranque dos trabalhos de análise e migração do Habilus/Citius para o Citius Plus. Portanto, após a proposta ter sido aprovada pelo Ministério da Justiça e a respectiva adjudicação ter sido efectuada.

À data do conhecimento daqueles documentos, uma possível análise "ad hoc" não adiantaria nada, uma vez que a proposta estava contratada e a auditoria tinha sido divulgada. Preferiu-se então aguardar pelo presente momento, para confrontar a proposta técnica, cujo conteúdo reproduz o essencial das auditorias, com o efectivo trabalho desenvolvido, no qual a equipa do ITIJ participou, quase exclusivamente enquanto fornecedora de conhecimento, facilitadora de logística administrativa e, sobretudo, executora da "motivante" tarefa de "beta tester" dos resultados finais de cada funcionalidade migrada, processos pelos quais já tinha passado enquanto "criadora" da solução original. A equipa do ITIJ sempre foi tratada como "cliente", termo a que não estava, nem está habituada…

Quanto à Auditoria, apenas se refere que é "*sui generis*", não há conhecimento de qualquer outra, numa actividade tão complexa e tão específica e num sistema informático com a dimensão do auditado, em que os auditores nem sequer tenham falado com a equipa auditada.

É notória a falta de clareza e destrinça entre o relatório de auditoria e a proposta de evolução tecnológica, performance e segurança, sem se perceber bem onde é que acaba uma e começa a outra. Ao contrário do que seria de esperar, a proposta técnica perde-se num labirinto de juízos de valor acerca do trabalho de terceiros, ao invés de se cingir exclusivamente à elaboração de um documento técnico, no qual é suposto, tão só, propor-se o desenvolvimento de uma solução tecnológica.

Na tabela seguinte são analisados alguns extractos da proposta tecnológica, que se consideram mais relevantes. Não se analisa a auditoria, uma vez que os aspectos mais importantes são realçados na proposta.

Extractos da Proposta Técnica da CSW	Comentários da Equipa do ITIJ
2.1 Actualmente, tendo o sistema já passado por vários estágios de desenvolvimento, denotam-se sérias dificuldades na implementação de evoluções, devido à falta de uniformização tecnológica entre os vários subsistemas. **Em alguns desses componentes, a tecnologia é realmente obsoleta, o que implica inclusive ausência de suporte por parte do fabricante** e dificuldades em obter conhecimento especializado para a implementação de novas soluções.	Uma aplicação que se encontra em produção há mais de 11 anos, sujeita a inúmeras alterações (decorrentes da evolução do "negócio", das solicitações dos utilizadores, das frequentes alterações legislativas e de várias iniciativas da tutela), apresenta alguns aspectos de natureza tecnológica menos actual e algumas funcionalidades menos conseguidas e/ou consolidadas. A frequência, o prazo e o âmbito das intervenções justificaram algumas decisões menos consensuais e menos amadurecidas por uma equipa constituía por um número muito reduzido de recursos. Contudo, a alegada "desactualização tecnológica" não condiciona o desempenho nem a fiabilidade da solução. O mesmo já não se pode dizer relativamente a algo que é uma característica dominante no CITIUS Plus: retrocompatibilidade com VB6. Esta abordagem, a partir da Framework 4 (tecnologia actual e utilizada no ITIJ), é considerada obsoleta por parte do fabricante (Microsoft): http://msdn.microsoft.com/en-us/library/microsoft.visualbasic.compatibility.vb6.support.aspx Constata-se também a utilização massiva, em todo o código fonte, de um "helper" da "Artinsoft" que funciona como uma espécie de emulador de VB6. Este artifício mantém, inclusive, grande parte da sintaxe nativa desta linguagem que, naturalmente, não existe em C#. O "helper" é usado para substituir a linguagem nativa de C# por instruções escritas na linguagem VB6, invocando depois uma camada de classes que as traduz para C#. Só entre estas duas abordagens, foram detectadas no código do CITIUS Plus, desenvolvido em C#, cerca de 10.000 referências ao termo "VB6".
2.2 Assume-se como objectivo do projecto descrito na presente proposta, dar o passo decisivo e estruturante no sentido da evolução tecnológica da plataforma CITIUS, de forma a permitir uma resposta mais eficiente e segura à gestão e tramitação processual judicial, bem como um incremento na qualidade e segurança do serviço prestado Nesse sentido pretende-se concretizar a substituição dos componentes obsoletos do sistema - componentes Habilus, suportados por tecnologia Visual Basic 6.	Com recurso a retrocompatibilidade com VB6, considerada obsoleta pelo fabricante, e com recurso ao "helper" da "Artinsoft", este objectivo dificilmente será atingido, para não se dizer, desde já, que será inatingível. Por razões que, obviamente, apenas se relacionaram com a poupança de recursos e com a facilidade da conversão automática, os componentes .Net seleccionados para substituírem os componentes .COM (componentes utilizados na tecnologia VB6), são os dos mesmos fabricantes, verificando-se em alguns (demasiados) casos, que os componentes .Net apresentam deficiências ao nível do funcionamento. Deficiências que não existiam nos correspondentes .COM. Esta situação está, agora na fase de piloto, a ser evidenciada pelos utilizadores.

Extractos da Proposta Técnica da CSW	Comentários da Equipa do ITIJ
2.2 … O trabalho a realizar, que será sempre executado de forma colaborativa com as várias equipas de desenvolvimento da plataforma CITIUS, terá os seguintes objectivos:	Nos trabalhos de conversão e de desenvolvimento relacionados directamente com o código fonte do CITIUS Plus, só intervieram equipas da CSW. À equipa do ITIJ foram reservadas as tarefas de efectuar a passagem do conhecimento que a CSW considerou necessário e de testar os resultados finais das conversões. O código fonte do CITIUS Plus foi disponibilizado à equipa do ITIJ em 20-04-2011 e os componentes necessários foram entregues em momento posterior. Os *timings* definidos e os recursos humanos disponíveis na equipa do ITIJ impedem, em tempo útil, uma análise aprofundada e completa. Seria necessário o reforço desta equipa e o envolvimento da CSW para "passagem do conhecimento" (no sentido inverso e com semelhante intensidade e abrangência).
2.2 … Reformular a infra-estrutura tecnológica da plataforma CITIUS, de modo a permitir a resposta mais eficiente às solicitações dos vários stakeholders, quer na capacidade evolutiva do sistema, quer no suporte às alterações legais;	Claramente este desígnio não foi alcançado: • Não existe acréscimo na capacidade evolutiva, uma vez que desta migração, terá resultado uma cópia de funcionalidades de uma tecnologia para outra. • A capacidade de resposta da equipa do ITIJ diminuiu significativamente, uma vez que não esteve directamente envolvida nos trabalhos de desenvolvimento do CITIUS Plus (para desenvolvimento ou reforço de competências) e repartiu o seu esforço entre o acompanhamento (passagem de conhecimento) do CITIUS Plus e o apoio à produção e manutenção correctiva do Hablus/CITIUS. • O CIITUS Plus é um projecto com várias abordagens e tecnologia diversa (prejudicado pelas características anteriormente referidas) que não apresenta evolução significativa sobre o Habilus/CITIUS. • O CIITUS Plus não é uma aplicação desenvolvida em VB6, mas também não apresenta características de uma aplicação tipicamente desenvolvida em C#.NET ou noutra linguagem de Programação Orientada a Objectos. É esmagadora a ausência dos típicos objectos de C# e, no seu lugar, é notória a permanência massiva de arrays e strings compostas com separadores, existentes no código fonte do Habilus/Citius, que, ao contrário de C#, não é desenvolvido numa linguagem orientada a objectos.

Extractos da Proposta Técnica da CSW	Comentários da Equipa do ITIJ
2.2 … Incrementar os níveis de qualidade, controlo e segurança do acesso à informação de negócio e garantir a auditabilidade dos respectivos acessos e acções;	Os níveis de qualidade e de controlo do acesso à informação de negócio são equiparados aos que existem na actual versão VB6. A arquitectura é a mesma. No capítulo da segurança, foram implementadas algumas recomendações da auditoria, acompanhadas, porque exequíveis, na versão VB6.
2.2 … Homogeneizar ambientes e soluções tecnológicas, de modo a retirar sinergias na exploração da plataforma;	Este objectivo ou espectativa conduz a várias interpretações. Referindo-se a outras soluções, complementares, como o CITIUS Web, CITIUS Mandatários ou Sistema de Gestão de Procedimentos de Injunção, não se verifica nenhum incremento de sinergias porque o nível de integração e/ou interoperabilidade é exactamente o mesmo que existe na versão VB6. Na perspectiva da própria aplicação e GUI dos diversos módulos, os ambientes não foram homogeneizados. Toda a conversão foi feita "AS-IS", ou seja, a interface gráfica do utilizador no CITIUS Plus é idêntica à do Habilus/CITIUS. Esta homogeneização fica prejudicada pelas questões referidas anteriormente, relacionadas com a utilização excessiva de retrocompatibilidade, wrappers e helpers, facilitadores para as conversões, mas tremendamente onerosos na manutenção e evolução.
2.2 … Introduzir práticas, ferramentas e procedimentos que servirão para suportar as actividades de desenvolvimento e incrementar os níveis de serviço e qualidade na posterior gestão e evolução da plataforma.	Para estas áreas, o ITIJ definiu como plataforma de desenvolvimento o Team Foundation Server 2010 e o Visual Studio 2010 (Team System 2010). Existem demasiados problemas de integração entre esta plataforma e a definida pela CSW, que por si só já apresenta problemas de ligação entre os componentes que a constituem: o Visual Studio 2008 não integra com o Redmine e/ou com o SVN, o Redmine não integra com o Hudson e com o SVN e o Hudson não integra com o Visual Studio 2008. Para atingir este objectivo será necessária uma migração, que desejavelmente possa consubstanciar um modelo de desenvolvimento adequado. Para que isso aconteça será necessário o desenvolvimento prévio de um *template* apropriado.

Extractos da Proposta Técnica da CSW	Comentários da Equipa do ITIJ
2.2 … Aumentar o grau de conhecimento sobre o sistema, através da descrição das principais funcionalidades e do seu comportamento, pela escrita de cenários de teste, consubstanciados numa especificação de testes de suporte à validação e aceitação.	Durante todo o processo de migração e desenvolvimento, a CSW apoiou-se totalmente na equipa do ITIJ para adquirir o conhecimento que na proposta afirma querer transmitir. O CITIUS Plus resulta de uma migração "AS-IS", do Habilus/CITIUS, desenvolvido pela equipa da DGAJ, agora do ITIJ, que conhece perfeitamente, a todos os níveis e graus o sistema e o negócio. Durante a fase de pré-produção, os testes e respectiva especificação foram alterados pela equipa do ITIJ para garantir que os mesmos seriam integrados e transversais a mais do que uma área de actividade do sistema. A proposta apresentada pela CSW era a mesma que havia sido implementada nas fases anteriores, com recurso a testes unitários.
2.2 … É importante ter em consideração que a presente proposta não visa efectuar qualquer alteração funcional, mudar o paradigma (por exemplo, substituir aplicações desktop por aplicações web) ou mesmo reorganizar as diversas fontes de informação.	Esta intenção foi respeitada. Manteve-se quase tudo como estava e, quando a tecnologia de destino da migração não o permitiu, foram inseridos alguns artifícios (Helpers, wrappers, retrocompatibilidade, etc.).
2.2 … Tais tarefas são relevantes mas, de modo a atingir os objectivos estratégicos de evolução do sistema, torna-se forçoso nesta primeira fase substituir os componentes actualmente suportados por tecnologias obsoletas, por tecnologias modernas e em alinhamento com a estratégia para a exploração de tecnologias e sistemas de informação. É convicção da Critical Software que após a execução das actividades contempladas no âmbito desta proposta, estar-se-á em condições adequadas de poder proceder às alterações funcionais e arquitecturais necessárias para garantir a evolução do sistema, tendo em consideração as linhas estratégicas traçadas para o CITIUS.	É convicção da equipa do ITIJ que a evolução do sistema não será possível sem se refazer todo o projecto relativo ao sistema de informação dos tribunais, incluindo a necessária mudança de arquitectura e de filosofia de funcionamento. A migração deste projecto, da tecnologia em que foi desenvolvido (Framework 3.5) para a tecnologia actual e utilizada nos projectos do ITIJ (Framework 4) pode apresentar problemas. A compilação da solução CITIUS Plus dá avisos de estar obsoleta, em virtude da utilização, consideravelmente extensa, de retrocompatibilidade com VB6. Também é nossa convicção (e da Microsoft: http://msdn.microsoft.com/en-us/library/ee839621.aspx) que na próxima versão da Framework, ou na seguinte, esta tecnologia deixará de ser suportada. Quando isso acontecer, o projecto CITIUS Plus ficará parado no tempo se, entretanto, não for apropriadamente refeito, de acordo com a tecnologia nativa da plataforma C#.NET.

Extractos da Proposta Técnica da CSW	Comentários da Equipa do ITIJ
2.3 … Criar e adaptar procedimentos automáticos sobre as ferramentas de conversão de código, tendo em conta as especificidades da organização cliente e o objecto de conversão, tal como se descreve em 3.1.3 Adaptação de ferramentas de conversão	Na nossa opinião os "helpers", "wrappers" e a retrocompatibilidade utilizados para conversão de código e mantidos posteriormente no mesmo, sem serem substituídos pelas funcionalidades nativas de C#, para além de serem um constrangimento ao nível do desenvolvimento, pelo necessário acréscimo de esforço na manutenção evolutiva e correctiva, são também contraproducentes e não parecem ser a abordagem correcta na perspectiva da integração de aplicações. São úteis apenas para conversões automáticas transitórias, devendo posteriormente ser substituídos por componentes nativos da Framework e, consequentemente, removidos do código fonte. No código fonte do CITIUS Plus foram detectadas as seguintes situações: • Conversor de instruções VB6 para instruções C#.NET. Ou seja, uma camada de *software* intermédia com que a aplicação tem que lidar, com os riscos que daí podem advir, quer na componente da manutenção correctiva, quer na componente da manutenção evolutiva. Os primeiros constrangimentos que se vislumbram são a imediata dependência do fornecedor do helper (Artinsoft) e a dificuldade de entendimento do código quando o mesmo for entregue e a equipa do ITIJ tiver que interagir com ele. • Retrocompatibilidade, como referido pelo fornecedor: http://msdn.microsoft.com/en-us/library/microsoft.visualbasic.compatibility.vb6.support.aspx e http://msdn.microsoft.com/en-us/library/ee839621.aspx
2.3 … Migração das aplicações Habilus cliente existentes, segundo a abordagem proposta em 3.2.2 Processo de Conversão. Esta actividade de migração contempla as melhorias inerentes a um processo deste tipo, como são pequenas optimizações de estrutura e performance do código, incluindo medidas de harmonização de base de dados, enquadradas no plano de projecto e esforço disponíveis. Logo, actividades estruturantes de harmonização de base de dados, código aplicacional e afins, não se encontram incluídas no âmbito da presente proposta, tendo por isso que ser objecto de proposta posterior;	Foram feitas as alterações estritamente necessárias para que a aplicação funcionasse na Framework 3.5. Em contrapartida foi introduzido o "helper" da Artinsoft" e retrocompatibilidade com VB6, que, dado o objectivo do projecto, deviam ter sido evitados a todo o custo. Também é evidente um aumento exponencial de duplicação de blocos de código, chega a ser dezenas e dezenas de vezes na mesma classe, em dezenas e dezenas de classes.

Extractos da Proposta Técnica da CSW	Comentários da Equipa do ITIJ
3.1.1 … A execução de um trabalho de auditoria, realizado pela Critical Software [AD-1], permitiu constatar diversas lacunas no sistema. Se por um lado essas lacunas derivam directamente das tecnologias envolvidas e nas características tecnológicas da arquitectura adoptada, por outro lado, derivam de um desfasamento das boas práticas de desenvolvimento de software tendo em consideração a complexidade e criticalidade do sistema em questão.	Grande parte do relatório da auditoria foi elaborado com base em código fonte desactualizado e em código fonte que nunca foi parte integrante das aplicações de e em produção.
4.3 … A presente proposta tem como âmbito a conversão dos controlos agora identificados e representados na tabela acima. Caso o código fonte a converter possua referências a outros controlos que não os agora identificados, a sua viabilidade de conversão necessita de ser analisada, em termos a acordar entre a CSW e o Cliente (oportunidade, prioridade, prazo, custo, entre outros) de forma a decidir da sua eventual concretização.	O código fonte do Habilus/CITIUS não continha referências a qualquer outro controlo e a equipa do ITIJ informou a CSW que, neste projecto (CITIUS Plus), não deveriam ser usados quaisquer controlos .COM. Curiosamente, a própria CSW injectou massivamente no projecto a retrocompatibilidade com VB6 e o "Helper" da "Artinsoft", que são completamente "estranhos" à linguagem nativa da plataforma que suporta o CITIUS Plus.

Extractos da Proposta Técnica da CSW	Comentários da Equipa do ITIJ
4.5 CORREÇCÃO DOS PROBLEMAS DE SEGURANCA Tal como foi anteriormente referido, a identificação de um conjunto de problemas de segurança deriva directamente da tecnologia empregue e das práticas utilizadas na codificação do sistema. Assim, no âmbito da conversão das aplicações, irão ser considerados várias correcções manuais que visam incrementar os níveis de segurança e que complementam as correcções automáticas da conversão. Tendo por base a informação que é conhecida pela Critical Software à data desta proposta, em particular na informação que se encontra documentada em [AD-1], irão ser corrigidas as seguintes evidências: ▪ Possível comprometimento das palavras passe dos utilizadores ▪ Injecção de SQL ▪ Uso generalizado das credenciais de administração de base de dados ▪ Comprometimento da integridade/confidencialidade de código executável ▪ Comprometimento da confidencialidade/integridade da ligação entre máquinas cliente e as bases de dados ▪ Execução de módulos do Habilus sem necessidade de efectuar a autenticação ▪ Acesso e modificação de gravações de audiências em Tribunais sem controlo de acesso	O relatório da auditoria refere palavras passe ao nível da aplicação CITIUS Mandatários. O Habilus/Citius tinha, e tem, mecanismos de defesa de injecção de SQL. O Citius Plus também converteu esse mecanismo. Durante a conversão foi encontrada uma única vulnerabilidade que foi corrigida. O Habilus/Citius não usa credenciais de administração. O tipo de compilação (nativa) do VB6 é diferente da compilação de .NET. Em VB6, ao contrário de .NET, não se consegue reverter o código executável para código fonte, alterá-lo e voltar a compilá-lo. A comunicação encriptada entre o cliente e o servidor foi implementada, quer no Habilus/CITIUS, quer no CITIUS Plus As credenciais de autenticação passadas entre módulos eram encriptadas e agora são duplamente encriptadas, tendo um prazo de validade de 20 segundos. A solução do CITIUS Plus consistiu na colocação de todo o projecto num único módulo executável, invocando a partir desse módulo DLL's. A equipa do ITIJ sempre evitou esta implementação devido aos problemas de performance que o VB6 poderia apresentar. Ainda está por confirmar se o CITIUS Plus vai passar incólume a esta questão. Para já, os requisitos mínimos de hardware exigidos para o CITIUS Plus são bastante superiores aos requisitos mínimos definidos para o Habilus/CITIUS. Os requisitos para acesso do Habilus/CITIUS aos ficheiros de áudio são e sempre foram os agora implementados para o CITIUS Plus que, aliás, se limitou a converter a funcionalidade. A resolução é ao nível dos procedimentos e monitorização e não a nível aplicacional. Para garantir a originalidade dos ficheiros áudio a equipa do ITIJ, ao nível do Habilus/CITIUS, vai implementar um algoritmo de *checksum* dos ficheiros de áudio. É desejável que esta funcionalidade seja também implementada no CITIUS Plus.

Extractos da Proposta Técnica da CSW	Comentários da Equipa do ITIJ
4.5 … Garantir o incremento do nível de segurança do sistema através da utilização de um salt no cálculo dos valores do hash. O salt não é mais do que um parâmetro adicional passado ao algoritmo de hashing cujo objectivo é eliminar a sua previsibilidade. Assim apenas é possível calcular o mesmo hash para uma dada palavra-chave se for conhecido osalt que originalmente foi utilizado para a sua criação. Esta técnica invalida a utilização de tabelas de descodificação disponibilizadas on-line uma vez que estas não conseguem prever o valor do salt definido. Promover a substituição manual dos vários comandos SQL por prepared statements. Este tópico encontra-se descrito na secção 4.6.1 Acesso a Dados. Garantir a criação de contas limitadas para os utilizadores da base de dados (alterando inclusivamente o nome de utilizador de "sa" para outro nome de utilizador menos frequente), de acordo com as necessidades específicas do Habilus.	Ambas as soluções, Habilus/CITIUS e CITIUS Plus, abandonaram esta proposta e passaram a validar-se directamente na AD.
4.5 … Garantir a implementação de uma ligação segura (cifrada) entre as aplicações cliente e a base de dados. Desta forma é impossível aceder ao conteúdo da comunicação, garantindo ao mesmo tempo a sua integridade. Para tal, será necessária a utilização de certificados, que deverão ser fornecidos pela actual equipa do ITIJ.	Implementado no Habilus/CITIUS e no CITIUS Plus.
4.5 … Incrementar o actual nível de segurança passando a utilizar métodos IPC para transmitir o testemunho de autenticação entre módulos.	As credenciais de autenticação passadas entre módulos eram encriptadas e agora são duplamente encriptadas, tendo um prazo de validade de 20 segundos. A solução do CITIUS Plus consistiu na colocação de todo o projecto num único módulo executável, invocando a partir desse módulo DLL's. A equipa do ITIJ sempre evitou esta implementação devido aos problemas de performance que o VB6 poderia apresentar. Ainda está por confirmar se o CITIUS Plus vai passar incólume a esta questão. Para já, os requisitos mínimos de hardware exigidos para o CITIUS Plus são bastante superiores aos requisitos mínimos definidos para o Habilus/CITIUS.

Extractos da Proposta Técnica da CSW	Comentários da Equipa do ITIJ
4.5 … A forma mais eficiente e segura de resolver definitivamente a situação em questão é integrar a autenticação do Habilus com a autenticação do domínio, num contexto de disseminação completa do uso da tecnologia Active Directory da Microsoft. Esta medida será analisada, após o arranque do projecto, tendo em consideração a sua viabilidade enquanto enquadrada no actual projecto de conversão, caso o esforço necessário para incorporar esta solução esteja alinhado com o actual esforço previsto.	Implementado no Habilus e no Citius Plus.
4.6.1 … De modo a garantir que não ocorrem problemas de injecção de SQL, todos os acessos a dados irão requerer intervenção manual da equipa de projecto, traduzindo os diversos comandos para prepared statements / bind variables. Adicionalmente, **será criada uma classe utilitária para lidar com aspectos comuns da gestão de ligações e da gestão de transacções. Esta classe permitirá uniformizar a abertura e fecho de ligações, o commit e rollback de transacções, proceder à recuperação de falhas e disponibilizar tradução de excepções baseadas no erro de ligação.**	Apesar de existir uma classe que supostamente uniformiza a abertura e fecho das ligações, a verdade é que existem, dispersas um pouco por todo o código, dezenas e dezenas de aberturas "*ad-hoc*" de ligações às bases de dados, na sua maioria com a mesma *connection string*, resultado da conversão de EXE's para DLL's, cujo código não foi adequadamente reformulado para, entre muitos outros aspectos, minimizar as ligações às bases de dados. Nesta vertente, aquilo que se verifica no código fonte do CITIUS Plus, é duplicação exagerada de código, centenas de vezes, relativa a "prepared statments". Também se verifica, massivamente, o uso do "helper" da Artinsoft para traduzir instruções oriundas de VB6 para C#.

Extractos da Proposta Técnica da CSW	Comentários da Equipa do ITIJ
4.6.4 Será ainda necessário despender esforço manual de conversão de código cujo controlo de fluxo e de excepções esteja suportado nesta forma, pois o código convertido automaticamente para estas situações poderá não ser facilmente perceptível, nem tirar partido das características da plataforma .NET, em particular no que diz respeito ao tratamento de excepções. Tentativamente, o código destino de um GoSub será extraído para um método individual, tornando mais entendível o código original. No entanto, é necessário ter em consideração que tal só será possível se esse bloco de GoSub tiver um retorno e não for utilizado como destino de Goto ou de On Error Goto. Durante a fase de análise será averiguada a hipótese de intervir sobre o código original, para remover estas chamadas, de modo a simplificar o processo de conversão.	O Habilus/CITIUS não usa, nem nunca usou, a instrução “GoSub”.
4.6.5 … Em alguns dos projectos analisados foi detectada a ausência de option explicit, o que significa que uma conversão literal implicaria a utilização de tipos variantes, com os impactos conhecidos quer ao nível dalegibilidade do código, quer ao nível do desempenho. Assim, estas situações deverão ser averiguadas caso a caso, durante a fase de análise, devendo-se tomar a decisão de intervir manualmente no código original (especificando o tipo de estruturas de dados) ou optar por manter a utilização de tipos variantes.	Em nenhum dos projectos integrantes do Habilus/CITIUS falta, ou alguma vez faltou, a declaração de “option explicit”. Nas práticas da equipa do ITIJ é, e sempre foi, obrigatório utilizar a declaração “Option Explicit”.
7.1. … O Plano de Garantia apresentado cessa após 24 (vinte e quatro) meses a contar da data de aceitação do produto/serviço. Todas as anomalias que tenham sido notificadas à Critical dentro do período de garantia são resolvidas mesmo que para além do período estabelecido. **O Plano de Garantia cessará, de imediato, nos casos em que existam alterações ao produto/serviço entregue não efectuadas pela Critical Software**.	A parte final desta cláusula é penalizadora para o MJ e, interpretada à letra, não permite que a equipa de desenvolvimento do ITIJ tenha qualquer intervenção, seja ela correctiva ou evolutiva sobre toda a solução entregue durante, pelo menos, 2 anos. Compromete todo e qualquer desenvolvimento deste projecto durante 2 anos, a não ser que seja efectuado pela própria CSW mediante adjudicação e respectivos custos associados.

Extractos da Proposta Técnica da CSW	Comentários da Equipa do ITIJ
7.2. … A **responsabilidade desta etapa é da Critical e do Cliente que devem trabalhar em conjunto para identificar a solução, mais eficaz e eficiente para a anomalia, no sentido de minimizar o impacto da mesma para o sistema e de minimizar o tempo e custos da sua resolução**. Implementação da Solução para a Anomalia. O objectivo desta etapa consiste no trabalho de implementação da solução para a anomalia, incluindo a actualização da documentação e as validações internas necessárias para garantir que a anomalia foi eliminada. Esta etapa termina com a entrega da solução ao Cliente. A responsabilidade desta etapa é da Critical, podendo existir necessidade de iterações com os responsáveis do Cliente para clarificar algum aspecto da solução. Validação da Solução. O objectivo desta etapa consiste na instalação dos materiais entregues pela Critical no Cliente e na validação, por parte do Cliente, que a anomalia foi resolvida. A responsabilidade desta etapa é do Cliente, podendo existir necessidade de iterações com os responsáveis da Critical para clarificar algum aspecto da validação da anomalia. Fecho da Anomalia. O objectivo desta etapa é garantir que a anomalia foi completamente removida do produto/serviço entregue pela Critical. Consiste na aceitação formal do fecho da anomalia. A responsabilidade desta etapa é da Critical e do Cliente que devem formalizar a resolução da anomalia. Caso a solução encontrada para resolver a anomalia tenha a necessidade de um tempo que não é compatível com o impacto que a mesma tem no sistema/negócio do Cliente, a Critical e o Cliente devem trabalhar numa Solução Temporária (workaround) para minimizar esse mesmo impacto até que a solução definitiva possa ser entregue e validada.	Sempre o cliente (através da equipa de desenvolvimento do ITIJ) a avançar primeiro ou solidário e com um envolvimento muito exigente. Quem é o responsável pelos problemas que a solução venha a apresentar?

Extractos da Proposta Técnica da CSW	Comentários da Equipa do ITIJ
7.3. O Plano de Garantia tem como compromisso que as anomalias imputadas à Critical na prestação do serviço descrito nesta proposta são resolvidas sem custos adicionais para o Cliente. Não é objectivo d oPlano de Garantia responder com SLA (ServiceLevelAgreements) à criticalidade e impacto que o serviço prestado possa ter no sistema/negócio do cliente, nos casos em que SLA são críticos para o negócio do cliente deve-se avaliar a necessidade de um Plano de Manutenção e Suporte. Devido a este facto, os tempos de resposta associados às etapas descritas no Plano de Garantia reflectem apenas o compromisso de solução das anomalias. A Tabela 25 apresenta os tempos indicativos associados a cada uma das etapas, sendo que existirá o empenho da Critical para, juntamente com o cliente, responder o menor tempo possível minimizando, ao máximo, os impactos da anomalia. Etapa Tempo de Resposta Reconhecimento da Anomalia 3 Dias úteis Implementação da Solução para a Anomalia 2 Semanas Validação da Solução 1 Semana Fecho da Anomalia 1 Semana	Inadequado por ser incompatível com a actividade dos Tribunais (negócio e *stakeholders*). A resolução de uma anomalia (desde o seu reconhecimento até à sua resolução definitiva) pode demorar mais de um mês.
8.3 … Em caso algum será a Critical Software responsável por danos indirectos incluindo lucros cessantes ou perda de dados.	Os danos podem resultar de deficiente implementação ou intervenção inadequada da CSW. Provando-se, de quem é a responsabilidade?

3. Análise do Código Fonte do CITIUS Plus

O Código fonte do CITIUS Plus foi disponibilizado em 20 de Abril de 2011. A equipa do ITIJ iniciou a sua análise no dia 26 de Abril de 2011.

Os componentes licenciados para o CITIUS Plus, indispensáveis para esta tarefa, foram disponibilizados posteriormente.

O objectivo de todo o projecto CITIUS Plus foi a migração do actual Habilus/CITIUS, que foi desenvolvido em Visual Basic 6, para uma nova tecnologia.

Para o efeito foi seleccionada uma empresa de desenvolvimento de *software* e escolhida como plataforma de desenvolvimento a Framework 3.5 da Microsoft e a linguagem de programação C#.

Atendendo à plataforma escolhida, vale a pena lembrar que a tecnologia seleccionada é baseada no paradigma da programação orientada a objectos, paradigma esse que tem vindo a ser constantemente melhorado, sendo cada vez mais funcional, objectivo e completo:
http://pt.wikipedia.org/wiki/Orienta%C3%A7%C3%A3o_a_objetos

O CITIUS Plus, que visa substituir o actual Habilus/CITIUS, só por ser desenvolvido numa plataforma tecnológica mais actualizada, robusta, fiável, flexível, etc., não resolve grande parte das limitações e problemas apontados à aplicação desenvolvida em VB6 e pode ter potenciado outros problemas e constrangimentos. Afinal, o que aconteceu foi uma simples conversão de uma linguagem de programação (VB6) para outra linguagem de programação (C#), mantendo-se a arquitectura e toda a filosofia de funcionamento anterior.

A conversão não corresponde às expectativas, uma vez que raramente são usadas as potencialidades da nova tecnologia. Exemplificativa é a total ausência de objectos criados pelo programador e partilhados entre os vários módulos da solução.

Numa aplicação desenvolvida em C#, surpreende a utilização do *namespace Microsft.VisualBasic*, que importa toda a funcionalidade do VB.Net, desnecessária em C#.Net. A análise é superficial, devido ao "*timing*" disponível, à escassez de recursos e à enorme complexidade do código fonte, pelo que não se consegue perceber qual o objectivo desta utilização. Crê-se que será necessária para possibilitar a utilização de alguns *artifícios* "injectados" na conversão para, de alguma forma, com base em "*workarounds*", se continuar a utilizar parte da tecnologia e/ou da filosofia e do comportamento do VB6.

Dos artifícios detectados, destacam-se dois *namespaces*, que, ao serem utilizados massivamente por todo o código migrado, inviabilizam qualquer tipo de intervenção que tenha como objectivo realinhar a solução para a utilização exclusiva de funcionalidades e de código nativo C#.Net:

- **Artinsoft.VB6.DB**: Um "Helper" desenvolvido pela Artinsoft, que emula o comportamento nativo de funções/métodos da tecnologia VB6 e os "mapeia" para C#. Grosso modo, a utilização deste "*wrapper*" obriga o programador, na sua normal actividade, a utilizar instruções da sintaxe de VB6 e, posteriormente, a um nível mais baixo, numa espécie de "*caixa negra*", esperar que a camada de software da Artinsoft faça o trabalho de tradução para a linguagem nativa de C# (ADO.Net).
  Este artifício representa, logo à primeira vista, dois constrangimentos:
  - Coloca o ITIJ e, consequentemente, todo o sistema de informação dos tribunais, na inteira dependência da Artinsoft, principalmente quanto a futuras e naturais evoluções da tecnologia;
  - Acrescenta imensas dificuldades nas tarefas de manutenção e evolução da solução. Neste caso, basta pensar-se, entre outras coisas, que será necessário a quem tiver

que interagir com o código fonte do CITIUS Plus, entender toda a sintaxe e funcionalidades associadas a este "helper" e depois saber como é que cada uma delas interagirá com a linguagem nativa de C#.

No próprio site da Artinsoft existem artigos que demonstram os riscos que se correm ao utilizar-se este subterfúgio. Por exemplo:
http://blogs.artinsoft.net/juan_fernando/archive/2007/11/16/1530.aspx no final do artigo pode ler-se: *"… **Naturally, strongly-typed and well-structured code is preferable**. However, this increases the amount of manual work that is needed to achieve Functional Equivalence*.". O que demonstra que esta opção apenas teve como preocupação a facilidade de implementação automática da conversão. Um projecto desta envergadura justifica a opção pela melhor abordagem, independentemente da necessidade de maior esforço e envolvimento de recursos humanos.

Como resultado desta conversão, temos a utilização de "*wrappers*" que simulam recursos de VB6 para usar em .NET, ficando assim um código fonte extremamente complexo e confuso, composto por C#, Visual Basic pelo namespace e "workarounds" pela utilização do software da ArtinSoft.

No fundo, foi implementado um "emulador" de um recordset, nativo de VB6, que não existe em C#, mas que no entanto é simulado pelo software da ArtinSoft. Imaginemos, por exemplo, que dentro de algum tempo, naturalmente acontecerá, a Microsoft implementa uma nova versão de ADO.NET (o software nativo de C#), com novos métodos, mais seguros, fiáveis, com maior desempenho, etc., do que a versão actual. O que acontecerá é que o ITIJ vai ter que esperar que a Artinsoft desenvolva um novo emulador (se desenvolver) para poder fazer, seguramente com custos desnecessários, o *upgrade* do CITIUS Plus.

Exemplo de utilização no código fonte:

```
ADORecordSetHelper rstLocal = new ADORecordSetHelper("");
```

Até o nome original da variável do tipo recordset usada em VB6 foi mantido: "rstLocal".

Acresce a tudo o que foi referido sobre a utilização deste "helper" da Artinsoft, o problema que é colocado ao nível do perfil de recursos a recrutar para manutenção correctiva e evolutiva da aplicação. Os recursos não podem ter apenas como competências C#.NET, também tem que ter competências em VB6, uma vez que grande parte dos mecanismos de funcionamento deste "wrapper" são baseados na sintaxe de VB6.

- **Microsoft.VisualBasic.Compatibility.VB6**: A utilização deste *namesapace* importa várias funcionalidades para "retrocompatibilidade" com VB6, situação de todo inaceitável, uma vez que desvirtua, por completo, o espírito da migração deste projecto. A utilização deste artifício significa que o CITIUS Plus não pode, ou não deve, depende do ponto de vista e do sentido de responsabilidade de quem tomar decisões, evoluir de plataforma. Ou seja, não pode ou não deve passar, por exemplo, da Framework 3.5 (onde foi desenvolvido o CITIUS Plus) para a Framework 4 (tecnologia actual).
  Estamos em 2011 e actualmente a tecnologia dominante é a Framework 4, Visual Studio 2010, tecnologia usada pelas equipas do ITIJ. Se for tentada a conversão do CITIUS Plus para esta versão da tecnologia e se for tentada uma compilação do código fonte, devido à utilização deste artifício, é apresentado um aviso que diz que a solução está obsoleta.

Estamos em crer e o fabricante (Microsoft) também (http://msdn.microsoft.com/en-us/library/ee839621.aspx), que este *namespace* numa futura versão da Framework deixará de ser suportado.

Ainda que se mantenha o CITIUS Plus eternamente baseado na Framework 3.5, nunca vai ser possível tirar partido dos processadores e Sistemas Operativos de 64 bits, uma vez que o CITIUS PLUS com este *namespace* só pode ser compilado para funcionar a 32 bits:
http://msdn.microsoft.com/en-us/library/microsoft.visualbasic.compatibility.vb6.support.aspx
e
http://support.microsoft.com/kb/972959/pt

Existe no CITIUS Plus uma solução completa, denominada "CitiusSync", que corresponde aos serviços de actualização, transferência e sincronização desenvolvidos em VB6. Quando estes serviços foram desenvolvidos a tecnologia não permitia o seu desenvolvimento como serviços Windows puros, tendo-se, na altura, recorrido a um "workaround" para os instalar no sistema operativo como serviços Windows. A migração do CITIUS Plus, apesar de neste momento na tecnologia da Framework existirem ferramentas para desenvolvimento de serviços Windows puros, manteve, precisamente, o mesmo "workaround", tendo os mesmos sido desenvolvidos como aplicações Windows Forms normalíssimas.

O código fonte do Citius Plus não está comentado. Os comentários que se encontram são os que existiam na versão VB6 e que foram automaticamente migrados, ou então comentários inseridos pela ferramenta de migração com indicações de problemas na conversão. Esta vertente foi ignorada. As boas práticas que serviriam *"para suportar as actividades de desenvolvimento e incrementar os níveis de serviço e qualidade na posterior gestão e evolução da plataforma*" (conforme referido no ponto 2.2 da proposta da CSW), foram pura e simplesmente ignoradas. Este objectivo não encontra suporte ao nível do código fonte, encontram-se classes e membros das classes sem qualquer tipo de comentário. A situação é visivelmente grave em membros de classe públicos, que deveriam ser comentados, aumentando o grau de conhecimento sobre o sistema, através da descrição das principais funcionalidades e do seu comportamento, bastando para isso utilizar algumas ferramentas disponíveis.

Numa verificação superficial, conseguiu-se perceber que não existiu grande cuidado relativamente à padronização, por exemplo, na declaração de variáveis. Esta situação em C# acrescenta dificuldades redobradas para quem interagir com o código fonte. Enquanto que VB6 é insensível, ou seja, se uma variável for declarada como "teste", quando for utilizada é sempre reconhecida, mesmo que se escreva "teste", "Teste", "TESTE", "testE". Em C# não é bem assim, uma vez que a tecnologia é sensível e cada uma das formas de escrita será entendida como uma variável diferente. A verdade é que foram encontradas diversas formas de declaração de variáveis no CITIUS Plus, incluindo variáveis declaradas com e sem assentos, com e sem caracteres especiais, etc..

Ainda ao nível da padronização, normalmente é aconselhado o uso das notações Camel Case e Pascal Case como padrões para as convenções, desaconselhando-se a notação Húngara em programação orientada a objectos. Ora não se verifica um padrão consistente e coerente ao longo do código fonte do CITIUS Plus, optando-se por manter, preferencialmente, os mesmos nomes das variáveis do código original em VB6. Situação indesejável por se tratar de tecnologias diferentes e com abordagens ao nível das convenções dos nomes igualmente diferentes.

Para uma melhor leitura e compreensão do código, é igualmente aconselhado que este seja organizado segundo um padrão. Verifica-se que são raras as classes ao longo do código que sigam qualquer tipo de padrão. Encontram-se imensas vezes membros privados misturados com membros públicos da classe. Esta situação não seria muito grave se a aplicação fosse desenvolvida por uma equipa que posteriormente fizesse a sua evolução e manutenção, o que, tudo indica, não será o caso.

Na tabela seguinte apresentam-se alguns exemplos e considerações acerca do código fonte do CITIUS Plus, na maioria das vezes desaconselhados pela biblioteca de patterns & practices (http://msdn.microsoft.com/en-us/library/ff921345.aspx), referida como documento de referência pela CSW.

EXEMPLOS DE CÓDIGO ILUSTRATIVOS E QUE MERECEM ALGUMAS CONSIDERAÇÕES

Código Fonte do CITIUS Plus	Comentários da Equipa do ITIJ

DbCommand dbCommand = **Artinsoft.VB6.DB**.AdoFactoryManager.CreateDbCommand(pDB);
 	
DbParameter dbParameter1 = dbCommand.CreateParameter();
dbParameter1.Direction = ParameterDirection.Input;
dbParameter1.ParameterName = "@dbParameter1";
dbParameter1.DbType = DbType.Int32;
dbParameter1.Value = pIDUtilRef.ToString();
dbCommand.Parameters.Add(dbParameter1);
	Este código é repetido dezenas e dezenas de vezes na mesma classe, e em dezenas e dezenas de classes, tendo como consequência um incremento descomunal do tamanho das mesmas e tornando o código fonte dos projectos extremamente difícil de compreender.
	Os parâmetros são criados sempre da mesma forma, com o mesmo nome, sempre imperceptível, e quase sempre com o mesmo tipo de dados.
	Esta situação podia e devia ser resolvida com um método genérico a ser invocado de cada vez que fosse necessário criar parâmetros. Esta simples preocupação em deixar o código fonte minimamente legível, arrumado e mais fácil de manter, pouparia cerca de 25.000 linhas de código.
	Acresce a esta dificuldade a dificuldade de compreensão e leitura dos parâmetros, uma vez que são sempre imperceptíveis, invariavelmente chamam-se "@dbParameter", seguido de um número, quando podiam e deviam identificar a operação a que se destinam. Esta situação repete-se ao longo do projecto cerca de 5.200 vezes.
	
	Patterns & pratices:
	
	"…If you have many classes that perform data access, you should think about consolidating repeated functionality into helper classes. Developers with varying levels of expertise and data access knowledge may unexpectedly take inconsistent approaches to data access, and inadvertently introduce performance and scalability issues.
	
	By consolidating critical data access code, you can focus your tuning efforts and have a single consistent approach to database connection management and data access..."

if (NullConv.ToDouble(rstLocal["IDQualidade"]) == 176)	Os valores fixos, numéricos ou não, continuam no código, como sempre estiveram, tornando agora as alterações bem mais complicadas, uma vez que quem conhece o negócio deixou de conhecer o código fonte e quem conhece o código fonte não conhece o negócio.
Processos(1, 0, pIDProcesso, "-1", "-1", "-1", "-1", "-1", "-1", "-1", "-1", "-1", "-1", "-1", "-1", pUtilizador.ToString(), "-1", "-1", "-1", "-1", "-1", "-1", vDetalhe.ToString(), vData.Trim(), "-1", "-1", "-1", "-1", "-1", "-1", "-1", "-1", "-1", "-1", "-1", pAprocessual, pIDProcesso, ref tempRefParam2, pBITrib, 0, pDB, "-1");	Toda a lógica se mantém como estava, disso é prova a linha ao lado, que se repete, nos mesmos termos, centenas e centenas de vezes. Este tipo de funções ou métodos poderia e deveria ser francamente melhorado e simplificado para ficar perceptível. Diga-se de passagem que a nova linguagem de programação (C#) até facilita, afinal é uma tecnologia orientada a objectos e estas situações são tipicamente candidatas a serem integradas num objecto.

Código Fonte do CITIUS Plus	Comentários da Equipa do ITIJ

switch (pBITrib)
 	
{
case 763:
case 495:
case 254:
case 292:
case 762:
case 232:
case 310:
case 480:
case 324:
case 482:
case 490:
case 509:
case 498:
case 496:
case 499:
flagGrava = false;
break;
default:
switch (pAprocessual)
{
case 14:
if (pStatusCS == 0)
{
flagGrava = false;
}
break;
case 2000:
case 2001:
case 2002:
case 2003:
case 13:
case 20:
case 23:
case 26:
case 30:
flagGrava = false;
break;
}
break;
}
if (flagGrava)
	Existem vários casos deste tipo, onde, mais uma vez, a prática está longe de ser a melhor.
	
	Poderiam poupar-se n linhas de código, multiplicando-as pelas vezes que estas situações aparecem (imensas mesmo).
	
	Facilmente se chegaria a uns milhares de linhas a menos.
	
	Neste caso ilustra-se o uso de um switch de forma manifestamente exagerada, que deveria ter sido substituido por um IF.

Código Fonte do CITIUS Plus	Comentários da Equipa do ITIJ

elseif (switchVar == 105)
{
vResp = vResp + pOrigem + "#Não foi possível estabelecer ligação com o servidor da Câmara de Solicitadores!";
}
elseif (switchVar == 111)
{
vResp = vResp + pOrigem + "#A informação recebida da Câmara de Solicitadores está incorrecta!";
}
elseif (switchVar == 200)
{
vResp = vResp + pOrigem + "#Ocorreu um erro inesperado na ligação ao WebService da Câmara de Solicitadores!";
}
elseif (switchVar == 1)
{
vResp = vResp + "Câmara de Solicitadores#O Solicitador de Execução indicado não está inscrito no Círculo Judicial da comarca do processo ou da comarca da citação/notificação!";
}
elseif (switchVar == 2)
{
vResp = vResp + "Câmara de Solicitadores#O Solicitador indicado não está inscrito, nem existem solicitadores de execução inscritos no Círculo Judicial da comarca do processo ou da comarca da citação/notificação!";
}
elseif (switchVar == 3)
{
vResp = vResp + "Câmara de Solicitadores#O Solicitador de Execução não pode ser desassociado porque já iniciou a tramitação do processo!";
}
elseif (switchVar == 101) {
vResp = vResp + "Câmara de Solicitadores#O processo já tem solicitador nomeado!";
}
elseif (switchVar == 102) {
vResp = vResp + "Câmara de Solicitadores#A identificação do processo está incorrecta!";
}
elseif (switchVar == 103) {
vResp = vResp + "Câmara de Solicitadores#A identificação do Tribunal está incorrecta!";
}
elseif (switchVar == 104) {
vResp = vResp + "Câmara de Solicitadores#O Tribunal indicado não consta da tabela da Câmara de Solicitadores!";
}
elseif (switchVar == 106)
{
elseif (switchVar == 108) {
vResp
}
…
	Encontram-se variadíssimas incoerências na construção do código. Pensa-se que por consequência da conversão automática e por falta de cuidado posterior na análise do resultado da conversão, uma vez que são detectadas numa análise bastante superficial.
	
	Estas incoerências podem e devem ser resolvidas. São aceitáveis quando se faz pela primeira vez o código e se corre contra o tempo com poucos recursos. Não deveria acontecer num projecto destes, com o objectivo de melhorar todo um sistema onde, supostamente, foi analisado o código para que o mesmo fosse optimizado.
	
	Neste caso, os IFs deveriam ter sido substituídos por um switch.
	
	Não parece que neste projecto tenha sido ponderada a optimização do código, concluindo-se que todo e qualquer problema que exista no actual código fonte do Habilus/CITIUS, se manterá também no CITIUS Plus.

Código Fonte do CITIUS Plus	Comentários da Equipa do ITIJ

usingVB6 = **Microsoft.VisualBasic.Compatibility.VB6**.Support;
 	
...
 	
vCB_NewIndex = vCB.Items.Add("Pendente");
 	
VB6.SetItemData(vCB, vCB_NewIndex, 0);
 	
vCB_NewIndex = vCB.Items.Add("Efectuada");
 	
VB6.SetItemData(vCB, vCB_NewIndex, 1);
 	
vCB_NewIndex = vCB.Items.Add("Adiada");
 	
VB6.SetItemData(vCB, vCB_NewIndex, 2);
 	
vCB_NewIndex = vCB.Items.Add("Continua...");
 	
VB6.SetItemData(vCB, vCB_NewIndex, 3);
 	
vCB_NewIndex = vCB.Items.Add("Anulada");
 	
VB6.SetItemData(vCB, vCB_NewIndex, 4);
	Existem inúmeros casos em que se recorre a VB6, situação que, obviamente, é inaceitável, obsoleta e até impensável:
	
	http://msdn.microsoft.com/en-us/library/microsoft.visualbasic.compatibility.vb6.support.setitemdata.aspx
	
	e
	
	http://msdn.microsoft.com/en-us/library/ee839621.aspx

publicstring get_CellText(int rowIndex, int columnIndex)
{
//VB6 ctList compatibility
if ((rowIndex < 0) || (rowIndex >= this.Items.Count))
{
//VB6 ctList return false and does not produce any error, in these cases.
returnString.Empty;
}
return get_CellText(this.Items[rowIndex], columnIndex);
}
	Esforço de programação para emular o comportamento de VB6, camuflando o comportamento real do componente .NET.
	
	Para a conversão ser efectivamente feita de VB6 para C#, isto não pode ser feito. Assim obriga quem for efectuar eventuais correcções, evoluções ou até simples análise de fluxo, a fazer um esforço redobrado para compreender aquilo que parece ser uma framework que emula comportamentos do VB6, camuflando as verdadeiras funcionalidades e/ou comportamentos dos componentes .NET.

Código Fonte do CITIUS Plus	Comentários da Equipa do ITIJ

private bool SendMsgHook(string nActo, byte pModuloOrigem)
{
bool result = false;
 	
if (pModuloOrigem == 4 || pModuloOrigem == 6)
{
Form rcv = this.MessageReceiver as Form;
if (rcv != null && !rcv.IsDisposed)
{
return MessageReceiver.ReceiveMessage(nActo);
}
}
 	
return result;
}
	A passagem de parâmetros entre módulos manteve-se. Aquilo que é passado continua a ser um array de strings, concatenados e separados por um caracter especial, sendo depois separados para serem usados. Neste caso, em C#, dizem as boas práticas que é "obrigatório" utilizar um objecto.
	
	Já os dados que são retornados, que antes faziam uso do sistema de mensagens do windows, agora são retornados através do uso de um interface para o efeito, para simular um evento.
	
	Os forms que recebem valores implementam o método ReceiveMessage que é invocado quando é necessário enviar informação.
	
	Uma abordagem fazendo uso de eventos seria do tipo:
	
	//Disparar o evento na classe que envia a mensagem
	public delegate void onReveiveInfo(object sender, EventArgs e);
	public event onReveiveInfo OnReceiveInfo;
	public virtual void RetornaInfo()
	{
	OnReceiveInfo(this, new EventArgs());
	}
	
	//Captar o evento quando acontece, e fazer o que tem a fazer
	…
	cls = new eventos.Class1();
	cls.OnReceiveInfo+=new eventos.Class1.onReveiveInfo(cls_OnReceiveInfo);
	...
	private void cls_OnReceiveInfo(object sender, EventArgs e)
	{
	textBox1.Text = e.ToString();
	}
	
	…
	
	Quanto a nós, o ideal em .NET para estas situações, seria mesmo fazer o uso de potencialidades completamente ignoradas, tais como delegates e criação de eventos.
	Os módulos que necessitavam da ConnectionString para se ligarem às Bases de Dados, continuam a recebê-la e a fazer ligações desnecessárias.
	
	Uma vez que deixaram de ser EXEs independentes e passaram a ser DLL's, não deveriam fazer novas ligações, mas sim usar a ligação existente no módulo a partir do qual são invocadas.
	Estas situações repetem-se dezenas e dezenas de vezes.

Código Fonte do CITIUS Plus	Comentários da Equipa do ITIJ

frmEntidadesMain frmEntidadesMain1 = new frmEntidadesMain();
using (var wrapper = this.GetSystemUtils(Globals.Unity.ManagedWrapper))
{
frmEntidadesMain1.Arguments = wrapper.Encripta("H@bilus", DU.UtilUser, 1) + "£4£" + i.ToString() + "£" + Text1[0].Text + "£" + DP.procIDProcesso.ToString() + "£0£" + aCargoInterv[9] + "£" + aCargoInterv[10] + "£" + aCargoInterv[4] + "£" + aCargoInterv[5] + "£" + aCargoInterv[6] + "£" + aCargoInterv[1] + "£" + aCargoInterv[0] + "£" + aCargoInterv[1] + "£" + aCargoInterv[6] + "£" + ctCombo1[2].get_ListData(ctCombo1[2].ListIndex).ToString() + "£" + DP.procCodTrib.ToString() + "£" + DP.procIDNacional + "£" + vCodTribCit + "£" + wrapper.Encripta(DU.UtilCon, DU.UtilUser, 1);
}
 	
frmEntidadesMain1.WindowState = FormWindowState.Normal;
frmEntidadesMain1.MessageReceiver = this;
 	
public bool ReceiveMessage(string msg)
{
int i = 0;
string[] aMsgIn = null;
try
{
if (msg != "")
{
aMsgIn = (string[])msg.Split('£');
double switchVar = Conversion.Val(aMsgIn[0]);
if (switchVar == 97)
	Na passagem de parâmetros para as DLL's existe uma variável do tipo String, chamada Arguments, que recebe uma string com separadores, que depois tem que ser analisada e separada para um array. Tal e qual como na tecnologia VB6…
	
	Existe neste caso a referência ao módulo de trás, o que não faz grande sentido, uma vez que na plataforma há, por exemplo, os eventos para evitar esta situação. Em VB6 estas situações eram "proibídas". Existia uma permanente preocupação com os recursos de hardware e o desempenho da aplicação.
	
	Neste caso em particular, um módulo pequeno e utilitário como é a gestão de entidades, tem a referência de um módulo gigantesco, como a secção de processos. O que demonstra que a preocupação com os recursos de hardware disponíveis e com o desempenho da solução pura e simplesmente não existe.
	
	Isto acontece em todos os casos de módulos que necessitem de trocar mensagens entre si.

private bool SendMsgHook(string nActo, byte pModuloOrigem)
{
bool result = false;
 	
if (pModuloOrigem == 4 || pModuloOrigem == 6)
{
Form rcv = this.MessageReceiver as Form;
if (rcv != null && !rcv.IsDisposed)
{
return MessageReceiver.ReceiveMessage(nActo);
}
}
 	
return result;
}
	Existem dezenas e dezenas de situações como esta, em que há variáveis declaradas que não servem rigorosamente para nada, como é o caso da variável denominada "result", que é declarada e inicializada com um valor e que não é alterada ao longo de todo o método.
	
	Mais uma vez se nota que a optimização de código, ou mesmo a "limpeza", não foi levada a cabo.

Código Fonte do CITIUS Plus	Comentários da Equipa do ITIJ

GC.Collect();
GC.WaitForPendingFinalizers();
	Esta invocação do Garbage Collector é usada 37 vezes no código. No entanto, no patterns & pratices, podemos ler o seguinte:
	"… Check that your code does not call GC.Collect explicitly. The garbage collector is self-tuning. By programmatically forcing a collection with this method, the chances are you hinder rather than improve performance.
	The garbage collector gains its efficiency by adopting a lazy approach to collection and delaying garbage collection until it is needed…"
	
	Por outro lado, em caso de absoluta e extraordinária necessidade, exemplifica o caso do seu uso:
	"…If you have a particular niche scenario where you have to call GC.Collect, consider the following:
	Call GC.WaitForPendingFinalizers after you call GC.Collect. This ensures that the current thread waits until finalizers for all objects are called.
	After the finalizers run, there are more dead objects (those that were just finalized) that need to be collected. One more call to GC.Collect collects the remaining dead objects.
	GC.Collect(); // This gets rid of the dead objects GC.WaitForPendingFinalizers(); // This waits for any finalizers to finish.
	System.GC.Collect(); // This releases the memory associated with the objects that were just finalized…"
	
	Como se pode ver, enquanto no patterns & pratices se aconselha a utilização cuidada, extraordinária e sequencial de 5 instruções para garantir a limpeza de todo o "lixo" da memória:
	GC.Collect();
	GC.WaitForPendingFinalizers();
	GC.Collect();
	GC.WaitForPendingFinalizers();
	System.GC.Collect();
	
	no código fonte do Citius Plus são apenas usadas duas instruções em 37 momentos distintos.

ADORecordSetHelper rstLocal = new ADORecordSetHelper(""); …	O uso do "helper" da ArtinSoft que emula os recordsets de VB6 cria sempre (absolutamente sempre, milhares de vezes) um dataset, mesmo quando só necessita de verificar se um determinado registo existe. Os datasets são óptimos para processarem várias tabelas em simultâneo, e para quando os dados são entregues aos controlos directamente, e não para quando se pretende processar um registo em concreto ou pequenas quantidades de informação, o que, na grande maioria dos casos é feito no CITIUS Plus. Para isso existem outro tipo de funcionalidades com um desempenho muito melhor: http://msdn.microsoft.com/en-us/library/8fxztkff(v=vs.71).aspx e http://msdn.microsoft.com/en-us/magazine/cc188717.aspx

Código Fonte do CITIUS Plus	Comentários da Equipa do ITIJ

object switchVar = rstLocal["Espécie"];
object switchVar = rstLocal["Modo"];
object switchVar_2 = rstAuxiliar["IDQualidade"];
private object HBDLL = null;
	Existe ao longo do código fonte do Citius Plus a utilização consideravelmente extensa de objectos genéricos, que a própria patterns & pratices desaconselha:
	"…Avoid using System.Object to access custom objects because this incurs the performance overhead of reflection. Use this approach only in situations where you cannot determine the type of an object at design time…"

using (var wrapper = this.GetSystemUtils(Globals.Unity.UnManagedWrapper))
{
wrapper.DesactivarBtnClose(this);
}
 	
using (var wrapper = this.GetSystemUtils(Globals.Unity.UnManagedWrapper))
{
r = wrapper.GetVolumeInformation(PathName, volname, volname.Capacity, out VolumeSN, out UnusedVal1, out UnusedVal2, fsname, fsname.Capacity);
}
	São desaconselhadas pela Microsoft as chamadas às APIs (Unmanaged code ou p/Invoke) sempre que possível: http://msdn.microsoft.com/en-us/library/ms973912.aspx
	No entanto, encontram-se dezenas de invocações que poderiam ser analisadas e evitadas, uma vez que os diversos módulos da aplicação deixaram de ser independentes.
	O código que se segue é um exemplo do uso da plataforma .NET, que evita uma chamada à API para desabilitar o botão "fechar" do form.
	`private const int CP_NOCLOSE_BUTTON = 0x200;`
	`protected override CreateParams CreateParams`
	`{`
	`get`
	`{`
	`CreateParams myCp = base.CreateParams;`
	`myCp.ClassStyle = myCp.ClassStyle | CP_NOCLOSE_BUTTON ;`
	`return myCp;`
	`}`
	`}`
	Neste caso, mais uma vez, seria de evitar a chamada à API (p/Invoke). A Microsoft explica como:
	http://msdn.microsoft.com/en-us/library/system.io.driveinfo.aspx

switch (Index) {
case 4:
if (Combo1[4].Items.Count > 0)
{
using (var wrapper = this.GetSystemUtils(Globals.Unity.UnManagedWrapper))
{
if (MouseButtons == MouseButtons.None)
Combo1[Index].DroppedDown = true;
}
}
break;
default:
using (var wrapper = this.GetSystemUtils(Globals.Unity.UnManagedWrapper))
{
if (MouseButtons == MouseButtons.None)
Combo1[Index].DroppedDown = true;
}
break;
} }
	Existem dezenas de casos onde é invocada uma classe que depois não é usada. Neste caso, em ambos os "cases" é usada a variável wrapper, que depois não tem qualquer aplicação no contexto. Além do mais não se justifica neste caso o uso de um switch, uma vez que o comportamento, seja qual for o "case", é sempre o mesmo.

Código Fonte do CITIUS Plus	Comentários da Equipa do ITIJ

(frmGProc.cs)
private void Combo1_KeyPress(Object eventSender, KeyPressEventArgs eventArgs)
{
var unmanagedWrapper = GetSystemUtils(ITIJ.Citius.Common.Globals.Globals.Unity.UnManagedWrapper);
var managedWrapper = GetSystemUtils();
	Também às dezenas, existem situações em que é inicializada uma variável no início do método que depois não é usada. Neste caso a "managedWrapper".
	
	Noutros casos (incluindo este), é inicializada também a variável "unmanagedWrapper" no início do método e só é usada num caso muito específico de um bloco pequeno de catch ou de else.

using (var wrapper = this.GetSystemUtils(ITIJ.Citius.Common.Globals.Globals.Unity.UnManagedWrapper))
{
vHWND = Convert.ToInt32(wrapper.FindWindowPartial(Globals.CitiusConstants.WindowTextMsExcel));
}
 	
using (var wrapper = this.GetSystemUtils(ITIJ.Citius.Common.Globals.Globals.Unity.UnManagedWrapper))
{
wrapper.ShowWindow(vHWND, Globals.Api32.SW_MINIMIZE);
}
	Neste exemplo é chamada uma API que é usada uma vez e na instrução imediatamente a seguir é chamada novamente a mesma API para ser usada de novo.

4. Conclusões

Como o fornecedor da solução (CSW) se propôs fazer, foi feita a migração da tecnologia VB6 para a tecnologia .Net (Framework 3.5).

É inegável, do ponto de vista do observador, que o Citius Plus se encontra a funcionar em ambiente piloto e, nesse ambiente, desempenha, com mais ou menos dificuldades, as mesmas tarefas que o antecessor desempenhava. Resta saber-se, objectivamente, quais são os custos associados ao modelo de migração seguido, que os autores do relatório consideram baseado na lei do menor esforço.

Quanto aos autores deste relatório, os custos associados a esta migração são demasiados quando comparados com os benefícios que a mesma poderá trazer. Estão amplamente referidos ao longo do relatório. Seguramente alguns desses custos serão incomportáveis ou mesmo inadmissíveis. O tempo, que se adivinha curto, o dirá.

O projecto CITIUS Plus implementou algumas funcionalidades adicionais para colmatar algumas situações relacionadas com questões de segurança referidas nos relatórios da auditoria. No Habilus/Citius, desenvolvido em tecnologia VB6, essas funcionalidades também foram implementadas. No entanto, neste novo projecto, perpetuam-se as fragilidades já existentes que constituíram fundamentos para a adjudicação e adiciona-se uma nova série de dificuldades e constrangimentos que devem ser ponderados. Destacam-se:

- Conhecimento do código fonte concentrado num grupo muito limitado de pessoas, que deixou de ser a equipa de desenvolvimento do Habilus/Citius e passou a ser a equipa de desenvolvimento da CSW, sendo que a situação se agravou, uma vez que esta última não tem quaisquer competências ao nível do negócio;

- Ausência total da necessária documentação referente ao projecto, particularmente de natureza técnica sobre a análise funcional e respectiva programação. Não existe sequer um diagrama de classes;

- Manutenção da arquitectura, manifestamente desadequada ao presente, que assenta numa filosofia cliente-servidor de duas camadas, convertida com ferramentas automáticas e, consequentemente, cegas quanto a questões como optimização, correcção ou reestruturação, mantendo-se o modelo de dados original, pensado e desenvolvido para uma realidade de há mais de 11 anos, quando ainda nem sequer existia rede judiciária. Ou seja, a fórmula da migração consistiu numa equação pouco ambiciosa e absolutamente confortável que se pode definir como “TO-BE = AS-IS”;

- Alheamento completo dos problemas actuais, muitos deles decorrentes da concentração de tribunais nas novas comarcas (NUTs). Problemas estruturais insolúveis na actual arquitectura, que não foi desenhada para a presente realidade e que só serão completa e definitivamente resolvidos com a alteração de arquitectura, acompanhada da correspondente alteração da filosofia de funcionamento das aplicações. Nesta matéria foi perdido cerca de um ano e meio e, atendendo aos últimos desenvolvimentos a este nível, urge solucionar a situação;

- Dificuldades acrescidas da equipa do ITIJ na gestão, manutenção e evolução da nova plataforma, que se sintetizam nos seguintes tópicos, amplamente referidos ao longo do presente relatório:

- Ausência da equipa do ITIJ dos desenvolvimentos relacionados com o código fonte do CITIUS PLUS;
- Escassez de recursos humanos na equipa do ITIJ;
- Código fonte completamente desajustado da linguagem de programação para que foi migrado;
- Utilização de componentes considerados obsoletos pelo próprio fabricante que, conscientemente, não permitem planear o natural objectivo de conversão da solução para plataformas tecnológicas mais actuais;
- Utilização de camadas de software de terceiros para facilitar a migração automática, que agora vão constituir um custo desproporcional para quem tiver que interagir com o código fonte;

Finalmente, face ao conteúdo do presente relatório, os autores sugerem, tendo em conta a necessária e desejável imparcialidade, que o ITIJ solicite a uma entidade externa e independente uma verificação objectiva que confirme, ou não, o exposto neste documento.