DISTRIBUTED SYSTEM FOR USABILITY TESTING

Tomáš Kubík

Master Degree Programme (2), FIT BUT

E-mail: xkubik15@stud.fit.vutbr.cz

Supervised by: Adam Herout

E-mail: herout@fit.vutbr.cz

Abstract: This paper describes the design of a software framework for usability testing. This extensive network framework and its protocol allow for integration of libraries for data collection from the basic peripherals like mouse, keyboard, camera, etc. If the protocol rules are implemented, these libraries can be platform independent. The client-server architecture allows for management of all collected data in a central database. The data in this database can be queried for evaluating the usability of applications.

Keywords: usability testing, framework design, user data collection

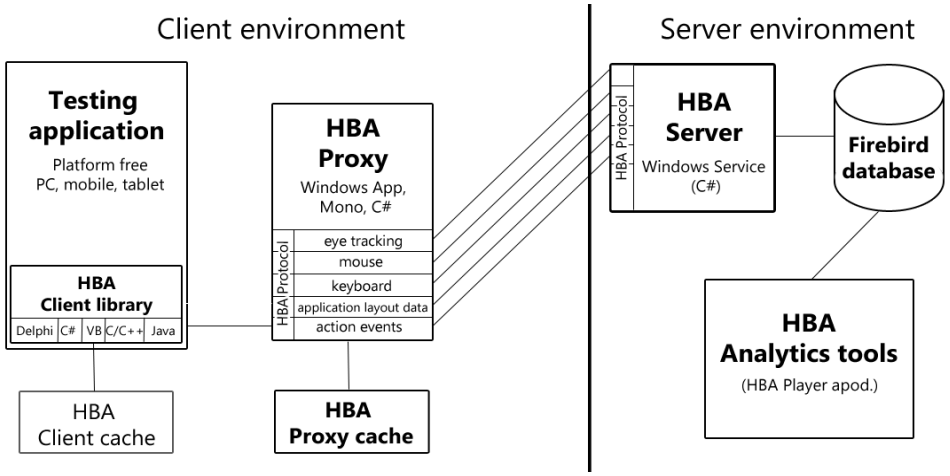
1. ÚVOD

Rozvoj grafického uživatelského rozhraní, kdy dnes není uživatel odkázán pouze na příkazový řádek, společně s tvorbou stále složitějších a propracovanějších aplikací má za následek to, že je uživatelské rozhraní jednou z nejdůležitějších částí celé aplikace. Ve snaze vytvořit intuitivní a snadno ovladatelné prostředí bez kompromisů, je jej nutné testovat přímo v reálném provozu. Tyto testy by měly napomoci k ohodnocení uživatelského rozhraní a zvýšit tak ergonomii a efektivitu celé práce.

Tento příspěvek popisuje navržený a implementovaný síťový framework HBA (*Human Behaviour Analytic*), jehož úkolem je sběr uživatelských dat z různých periférií (klávesnice, myš atd.) nezávisle na pozici (firma, domov apod.) a následné umožnění využití těchto dat pro vyhodnocení chování uživatelů. Tato data jsou ukládána centrálně v databázi, kde je možné s nimi pracovat. HBA framework je navržen tak, aby jej bylo možné použít na mnoha aplikacích různých platforem.

2. STRUKTURA HBA FRAMEWORKU

Framework využívá klient-server architektury a rozložení je tak situováno na 2 stěžejní celky, které blíže popisuje obrázek 1. Popis komponent systému je součástí dalších kapitol.



Obrázek 1 : Struktura HBA frameworku

2.1. HBA PROTOCOL

Veškerá komunikace ve frameworku probíhá v rámci kanálů, které jsou navržené nad protokolem TCP/IP [1]. Ve finále je jeden kanál vyhrazený pro data z klávesnice, druhý pro data z myši apod. Data odesílaná prostřednictvím těchto kanálů jsou z důvodu úspory datového toku binární a v současné implementaci umožňují i kompresi. Díky této struktuře je možné v budoucnu bez větších problémů přidat další kanály (např. „touch“ kanál). Při zahájení komunikace mezi klientem a serverem (případně proxy) je vyžadován triviální *handshake* díky kterému je možné předat klientu důležité informace a celkově tak zajistit vzájemnou kompatibilitu. Součástí HBA protokolu je i definice jednotlivých formátů binárních dat při síťové komunikaci.

2.2. HBA CLIENT LIBRARY

Tato knihovna je součástí testovaného systému a předpokládá se, že bude z důvodu kompatibility a určitého komfortu psána ve stejném jazyce jako testovaný systém. Předejde se tak různým problémům v přístupu k informacím o komponentách testované aplikace apod. Účelem knihovny je přihlášení aplikace do celého systému a poskytování minimálně základních dat. Sama knihovna tedy systému předává informace o testované aplikaci (verze, instance atd.) a základní datové kanály jako akce, informace o rozložení formulářů apod. Pravidla a formát dat při předávání těchto informací definuje HBA protokol. Současná implementace podporuje Delphi a C# (WinForms) a v budoucnu mohou přibýt další (WPF, Visual Basic, QT/C++ apod.).

2.3. HBA PROXY

Důvodů k existenci HBA Proxy je hned několik. První z nich je snaha implementačně odlehčit vývojářům testované aplikace, pokud pro využití HBA frameworku budou muset psát svoji klientskou HBA knihovnu. V této fázi mohou vývojáři implementovat pouze základní funkcionalitu a nést tak menší rizika spojená např. s pádem aplikace v závislosti na kódu, který do aplikace vlastně vůbec patřit nemusí (nemluvě o finanční stránce věci). O hodně věcí se může postarat právě proxy, která vlastně přihlašuje testované aplikace k odběru svých kanálů. Samotná proxy je pro klienta „neviditelná“. Klient tak nepozná, zda je připojený k proxy nebo přímo k serveru, a není tak nutná žádná konfigurace navíc. Dalším důvodem pro využití je správa dat (*HBA Proxy cache*), kdy proxy pozná, zda je připojení k serveru dostupné a v opačném případě začne data zálohovat na disk, kde čekají na další spojení. Díky takto oddělené aplikaci mohou být data odesílána i ve chvíli, kdy už testovaná aplikace neběží. Posledním důvodem může být i bezpečnost, kdy pád proxy, jakožto samostatné aplikace, nijak neohrozí běh testovaného systému a uživatel tak nemusí nic poznat.

2.4. HBA SERVER

Server je implementován jako systémová služba, u které není nutné, aby běžela na mnoha platformách. Z toho důvodu byl zvolen jazyk C# a platforma Windows [2]. Úkolem serveru je shromažďování a bufferování (dočasné ukládání) příchozích dat, které jsou po validaci, případné dekomprimaci a dekódování postupně ukládány do databáze. Server podporuje současné zpracování více příchozích požadavků.

2.5. DATABÁZE

K uložení dat je využita relační databáze nad volně šiřitelném databázovém enginu Firebird [3]. Hlavním důvodem k výběru této databáze jsou profesionální vlastnosti umožňující kvalitní vývoj databázových aplikací (uložené procedury, triggery, dotazy z procedur, domény, UDF funkce apod.). Velkým plus je i dostupnost kvalitních nástrojů pro správu databází pod tímto DB strojem.

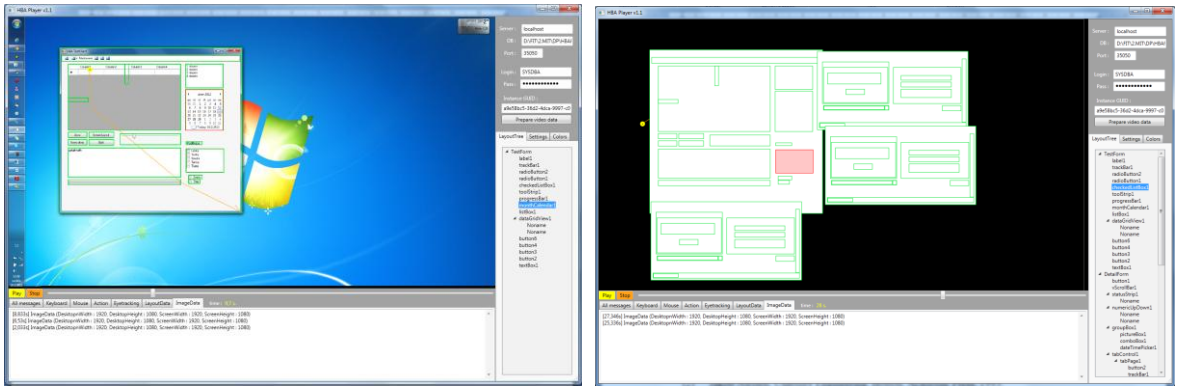
3. ANALÝZA ZÍSKANÝCH DAT

Po implementaci HBA frameworku do testovaného systému a získání prvních dat je nutné tato data zkontrolovat a zjistit, zda vývojářům umožní pokládat dotazy (tvořit analytické nástroje) tak, jak by

si přáli. Jedná se o kontrolu správnosti zasílaných rozměrů formulářů a komponent, kontrolu souslednosti obrazových dat s layout daty apod. Pro tuto validaci, a nejen tu, byl vytvořen HBA Player.

3.1. HBA PLAYER

Tato aplikace se připojuje přímo k databázi a stahuje veškerá data k určité instanci. Instance je v tomto případě časově omezený běh programu dostupný pod svým ID. Z těchto dat se vytvoří časová linie, kterou je možné přehrát od začátku do konce, nebo jednoduše pohybovat z místa na místo. Player v každém okamžiku rekonstruuje to, co uživatel s testovanou aplikací dělal (tj. zobrazí pohyb myši a její stopu, obrazová data, layout data, ve zvláštních oknech zobrazuje akce, hierarchii komponent apod.). Tuto aplikaci lze použít jak na zmíněnou validaci získaných dat, tak i na studii uživatelů, kteří v budoucím vyhodnocení vybočí z normálu.



Obrázek 2 : V levém okně je k vidění aplikace HBA Player se zobrazenými všemi kanály. V okně napravo je stejná aplikace se zobrazenými layout daty a označenou vybranou komponentou.

4. ZÁVĚR

V současné době je systém nasazen v rozsáhlém ekonomickém informačním systému a probíhá validace získaných dat. V blízké době se pak plánují zátěžové testy na více uživatelích a první vyhodnocovací dotazy nad kritickými částmi systému, kde je kladen důraz na efektivitu práce a vysoký komfort pro uživatele, kteří s aplikací přicházejí denně do styku. Očekává se, že integrace tohoto frameworku do testované aplikace otevře oči návrhářům a analytikům a umožní tak navrhovat uživatelské rozhraní mnohem rychleji, přesněji, levněji a hlavně s ohledem na zvyky samotných uživatelů. Skoro každá větší firma podobné nástroje má, bohužel jsou většinou přizpůsobené na daný produkt, nebo jsou součástí nedostupného firemního *know-how*. Tento framework bude, jako jeden z mála, veřejnosti dostupný a díky stávajícímu návrhu i snadno rozšiřitelný o další funkcionalitu.

REFERENCE

[1] John C. Snader, Effective TCP/IP Programming: 44 Tips to Improve Your Network Programs, Addison Wesley Professional, 2000

[2] Jay Glynn, Karli Watson, Bill Evjen, Morgan Skinner,Christian Nagel., C# 2008 – Programujeme profesionálně, Computer Press, 2009.

[3] Pavel Císař, Interbase/Firebird – tvorba, administrace a programování databází Boston, Computer Press, 2003