Blaise NG Architecture

IBUC 2010 – Pre Conference workshop

Introduction

- Why a new version?
- Design principles
- New technology
- Layout
- Runtime
- Deployment
- Current state (Phase 2)

Why a new version?

- Blaise 4.x code technically outdated:
  - Written in Delphi, VB6, C++
  - Originally designed for 80's & 90's hardware:
    - Single machine approach
    - Focus on low memory usage
    - Data storage file-based
  - Hard to implement new features:
  - Fundamental redesign needed

New architecture

- Blaise Next Generation White Paper (2007):
  - Design for concurrent use
  - Flexible Layout
  - Data Storage in RDBMS
  - Prepare for Language Enhancements:
    - Indefinite arrays
    - Randomness and rotation
    - User-defined Field attributes
    - .....
- Opportunities:
  - Correct flaws in Blaise 4.x
  - Use latest technology

Design principles

- Maximize common definitions:
  - Implementation of Blaise:
    - One general layout framework
    - One general runtime model
  - Blaise users:
    - One definition for multimode survey
- Separation of concern:
  - Use separate objects/services for each aspect of Blaise functionality
- Maximize machine independance:
  - Allow to run survey on arbitrary machine

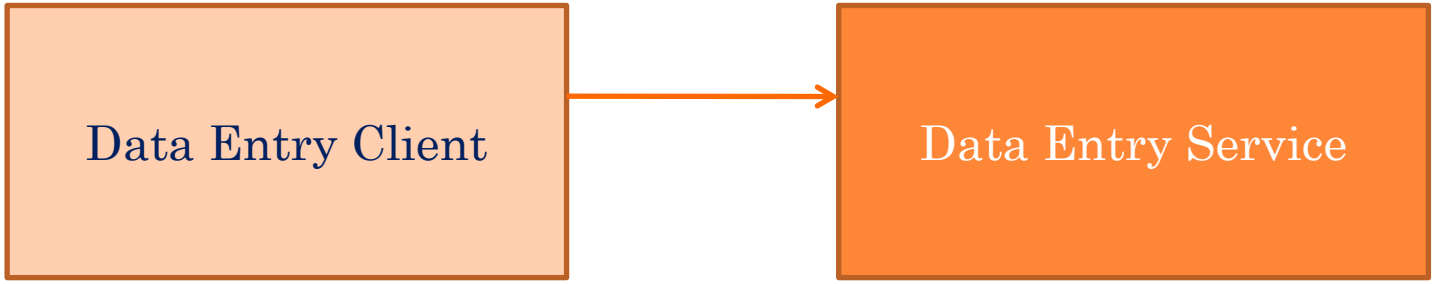
New technology

- Blaise NG written in C# (.Net Framework)
  - Unicode texts
- Layout based on Windows Presentation Foundation (WPF):
  - Separation of appearance and behaviour
- Runtime based on Windows Communication Foundation (WCF):
  - Service Oriented Applications
  - Interoperability by Messaging over Network

Layout

- Based on templates
- Controls:
  - Appearance
  - Events
- Bind to Blaise Objects:
  - Data Field
  - Interview State
  - Parallel
- WYSIWYG layout designer
- Flexible: design your own templates

Runtime

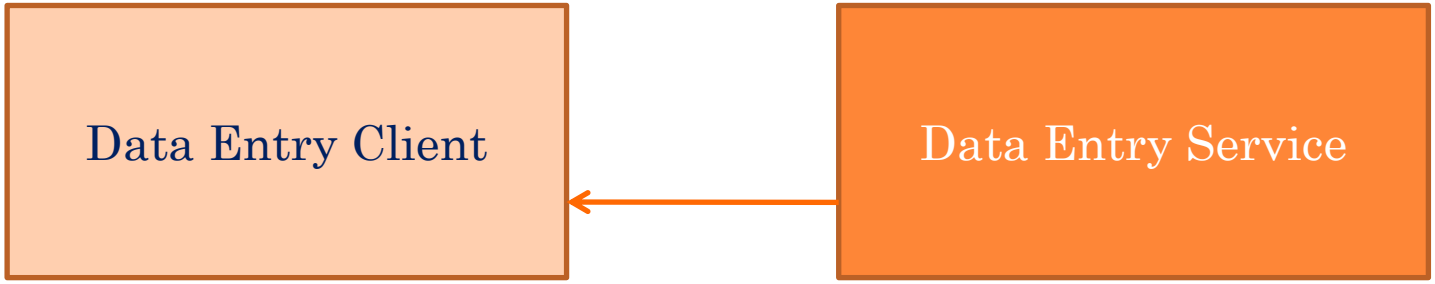


1. Data Entry Client submits data and action

Runtime

1. Data Entry Client submits data and action
2. **Data Entry Service updates session state**

Runtime



1. Data Entry Client submits data and action
2. Data Entry Service updates session state
3. **Data Entry Service returns new page**

Runtime

1. Data Entry Client submits data and action
2. Data Entry Service updates session state
3. Data Entry Service returns new page
4. **Data Entry Client shows new page**

Data Entry Client

- User interface for Interview
- Performs range checking
- Does NOT execute rules
- Updates state through Data Entry Service
- Two versions:
  - Windows Client (WPF)
  - Internet plugin (Silverlight)

Data Entry Service

- Provides Blaise functionality:
  - Executes rules
  - Maintains interview state
  - Creates interview page
- Efficient concurrent use:
  - Scalable
  - Stateless
- Session Database:
  - Resume interview without delay

Rules Engine

- Design goals:
  - Compatible with Blaise 4.x
    - Exception: correct flaws
  - Run independant of machine (settings)
- Data Entry Settings:
  - Influence execution
- Selective Checking Mechanism:
  - Optimises performance
- To do: test complex datamodels

Instrument Deployment

- Data Entry Client runs deployed instruments
- Deployment:
  - Register instrument name
  - Copy required files to specified locations:
    - Prepared datamodel
    - Layout resources
    - External data
- Currently:
  - Single machine only
  - Out of the box

Current state (Phase 2)

- New implementation:
  - Meta:
    - Nearly complete: test your complex datamodels
  - Layout:
    - Only for single questions (Field Panes)
  - Data Storage:
    - Relational database
  - Runtime:
    - Rules engine: almost complete
    - Data Entry Client: basic functionality for running interviews
  - Deployment:
    - Single machine only