penrillian™

Supporting Many Platforms

Making Your Killer App Dominate the Mobile World

UIQ TECHNOLOGY

www.penrillian.com

FORUM NOKIA

symbian Platinum Partner

Overview

- Three key developer techniques
- Portable architectures
- Two worst mistakes

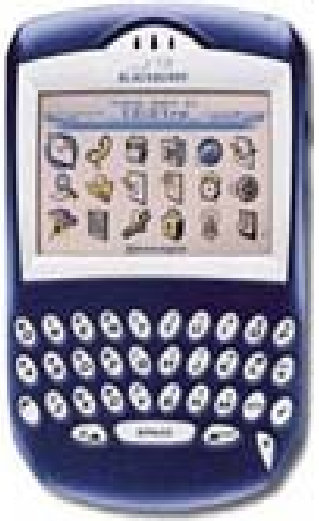
But first… Why port rather than rewriting?

Common ‘business logic’

Thousands of functionality ‘decisions’…

- “If it’s Thursday and it’s raining…”
- “Retry network connections 5 times, at intervals 10s, 11s, 15s”
- “User presses cancel, then retry, twice…”

Almost impossible to reimplement from specification.



And…

Projects that do a complete rewrite…

…Tend to fail!

- C.f. Chad Fowler ‘big rewrite’ etc.

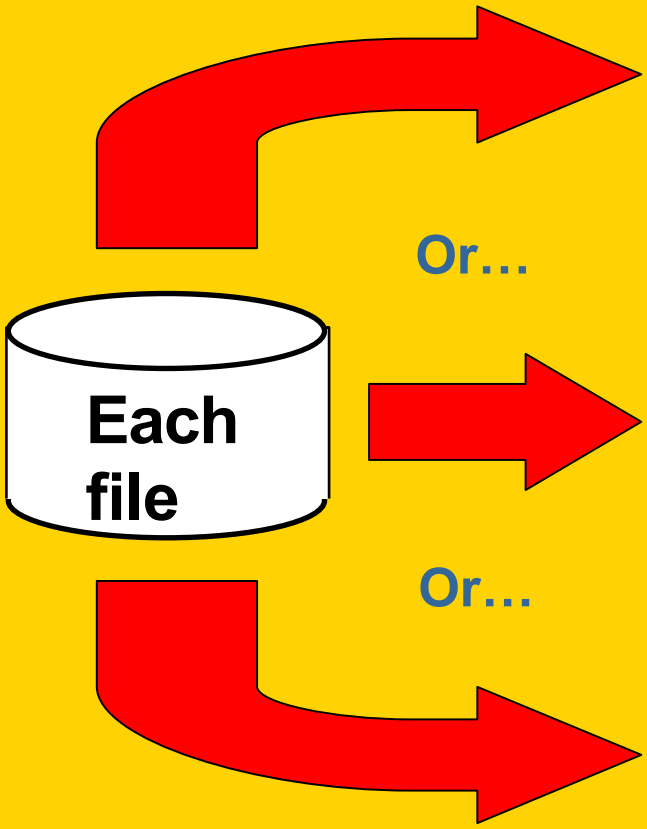
Three key techniques

1. Code triage

3. Refactoring to produce portable code

5. Test-driven porting

Code triage



Portable code

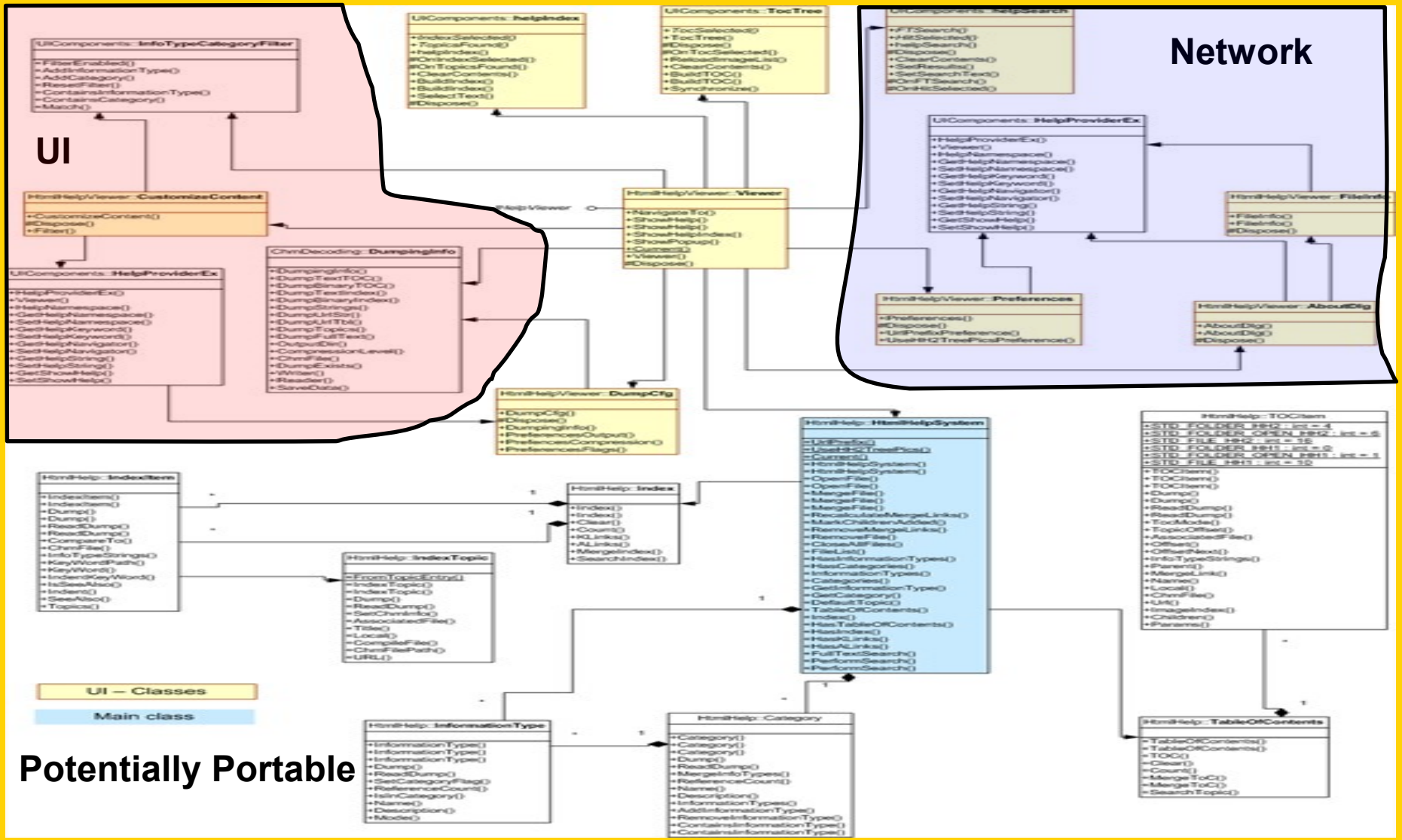
Potentially portable code

- Would be portable if refactored

Platform-specific code

- Not portable and never will be

Architectural decisions



Refactoring code

Some code is immediately portable

- **Great! Just recompile**

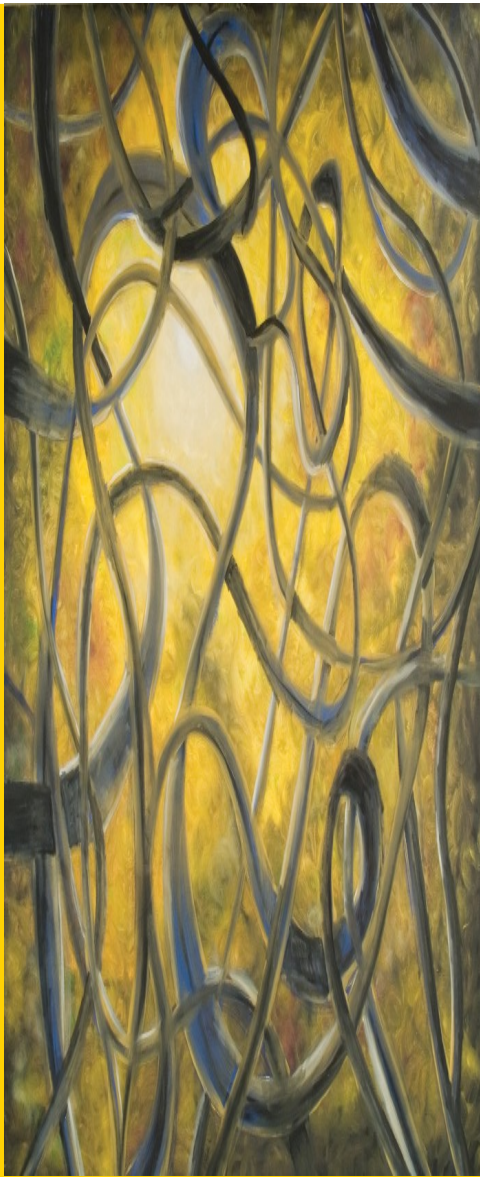
Much code could be portable:

- **Business logic**
- **Rendering code**

But beware entanglement…

- **UI event handlers that do business logic**
- **Comms interfaces not abstracted**
- **Rendering code not parameterised**

This code needs refactoring.



Refactoring danger

Changing code introduces…

…bugs!

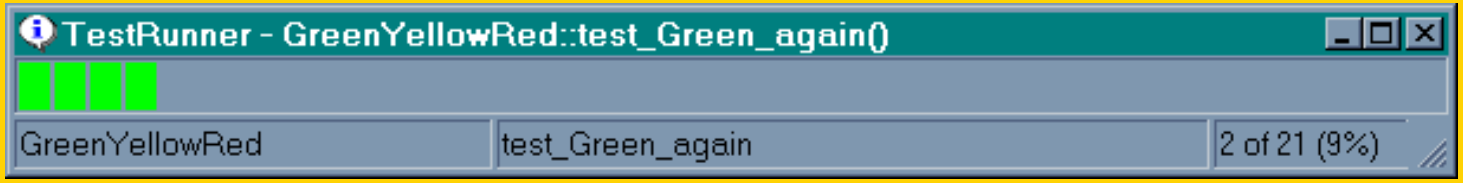
Solution: Test-Driven Porting

Refactor just enough to get it under test on the original platform.

- Don’t need a full test set
- A few ‘involve everything’ tests will get you going
- Use logging in the application to get test data.

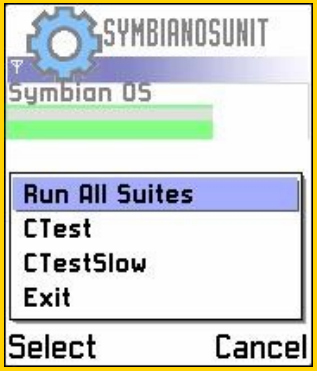
```
struct RequestAndResponse  {
    const char* request;
    const char* response;
};

void EndToEndTests::testSampleCommand()
{
RequestAndResponse test =  {
        "POST%20%2Fsample%3Fcommand%5F%20HTTP%2F1%2E1%0D%0AHost%3A…",
        "HTTP%2F1%2E1%20200%20OK%0D%0ADate%3A%20Wed%2C%2025%... " };
    checkHttpRequest( test );
}
```

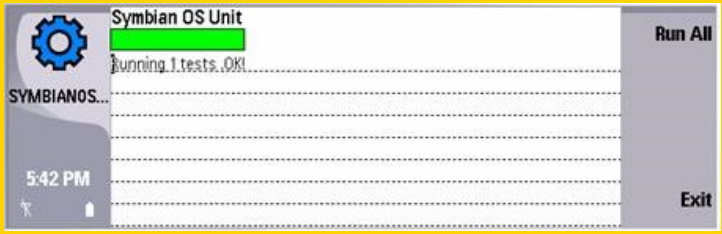
Introducing CxxUnit



Arguably the best of the C++ unit test frameworks…

- SymbianOSUnit Supports all Symbian OS UI flavours
- Other versions: MS Windows, PPC, Palm etc.





Steps to a shared engine

Case Study: Migrating a C++ Engine

1. Get the end-to-end tests running on source
   - Now you've repeatable tests
2. Dummy out communications layers and similar
   - You've removed the external dependencies
3. Façade out non-portable libraries
   - Code is now portable
4. Now get it running on the target…
   - You've a ported engine on the target
5. Add UI, comms layers etc.

1. Dummy out UI Callbacks

This was the most dangerous bit…

```
class MockApplicationUI:
        public CMainEngine::MObserver
{
public:
         MockApplicationUI();
        void OnEncryptionKeyRequested() {}
        void OnEncryptionKeyResponse(TEncryptionKeyResult
aResult) {}
        void OnConnect(const char* aDeviceName) {}
        void OnDisconnect(bool aFirstConnectionAttemptFailure) {}
…
```

('Extract Interface…')

2. Dummy out comms

Extract interface:

- Abstract base class keeps the name;
- Implementations called '…Impl', 'Mock…', '…Win32'

```
class MockCommServer :
        public CommServer
{
public:
        MockCommServer(void);
        ~MockCommServer(void);
        virtual unsigned long DoConnect(GPRSConnection*
connection, std::string username, std::string password) { return 0; };
        virtual void Connect(CommServer::ConnectType) {};
        virtual void Disconnect(bool aTryToConnectAgain) {};
        virtual void IssueSend(std::string aData);
        virtual void IssueReceive();
…
};
```

…Dummy out comms

Use logging (URL encoded) to get values…

- Keep logging statements, so tests can change as needed

```
struct RequestAndResponse  {   const char* request; const char* response; };

static RequestAndResponse CorrectResponses[] = {
   { “AA%3AAQ00000000",    “667%3C%3Fxml%20version%3D%22…” }  };

static string lastRequest;
void MockCommServer::IssueSend(std::string aData)
{ lastRequest = aData; }

void MockCommServer::IssueReceive()
{
   string patternToFind = Utilities::UrlEncode( lastRequest );
   int i;
   for (i=0; patternToFind != CorrectResponses[i].request; i++) {
      assert( i<(sizeof(CorrectResponses)/sizeof(*CorrectResponses));
   }
   string reply = Utilities::UrlDecode( CorrectResponses[i].response );
   m_Observer->HandleReceivedData( reply );
}
```

3. Façade out libraries

```
class Semaphore  {
public:
        Semaphore();
        ~Semaphore()
        /** Blocking up to aTimeout ms until semaphore available */
        bool WaitFor( int aTimeout );
        /** Release semaphore*/
        void Release();
};
```

with different implementations for each platform:

```
class Semaphore  {     // Windows semaphore…
Semaphore() {
      m_ResultSemaphore = ::CreateSemaphore(NULL, 0, 1, _T("")); }
   // ..etc

    HANDLE m_ResultSemaphore;
};
```

No need for ‘extract interface’!

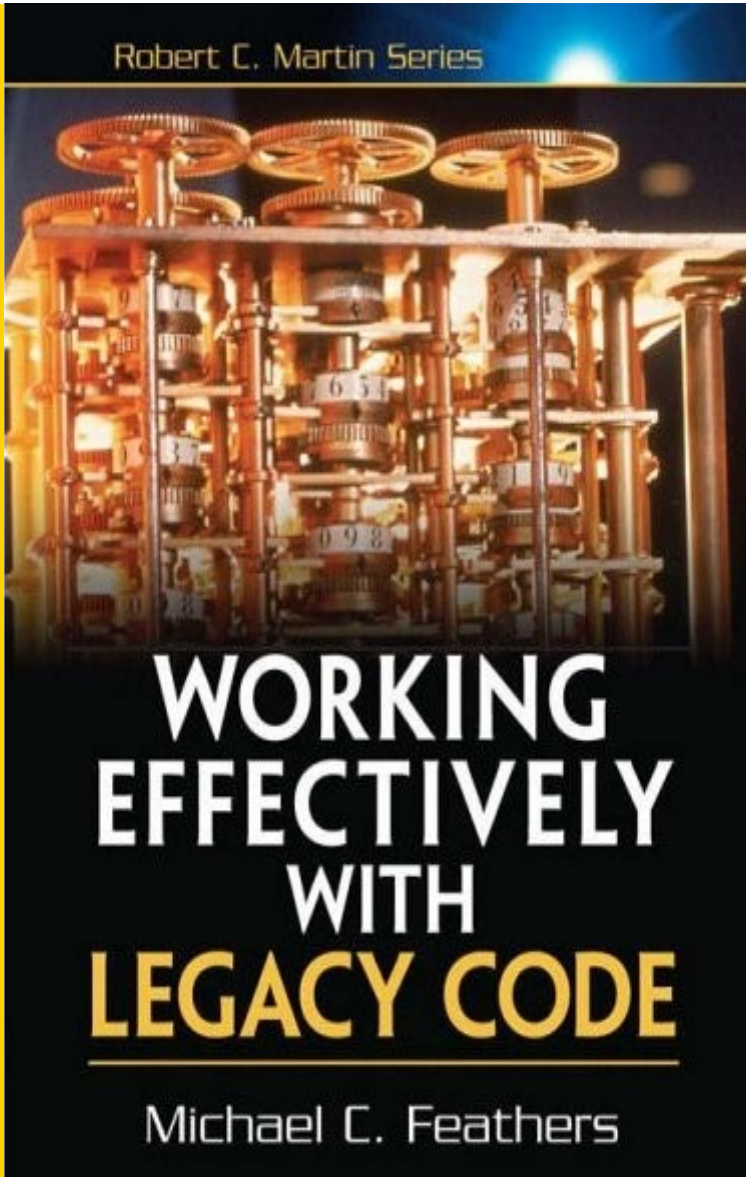
How to do it better…

Legacy Code,
by Mike Feathers

- Legacy = ‘not yet under test’!

Don’t try to read it through!

- For this, skim the intro chapters, then skim part 3; last bits are the most useful.



Libraries…

Where possible, reuse

- SourceForge
- External suppliers
- uSTL for Symbian OS
- Libraries built on POSIX

Or emulate functionality of source platform

- E.g. Minimum string class?

4. Running on the target

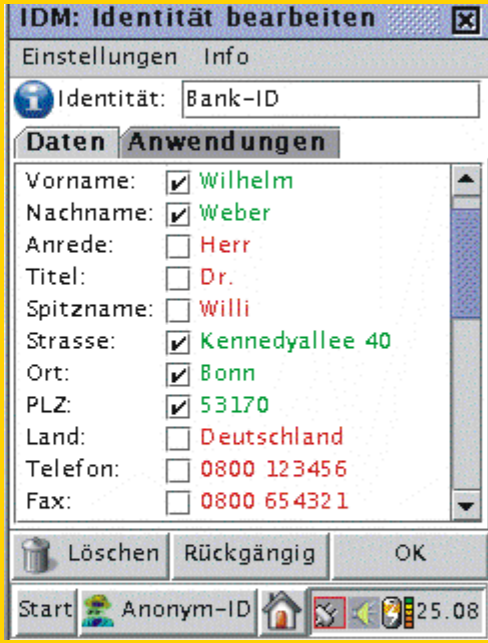
Get the portable engine … ported!

This is the easy bit…

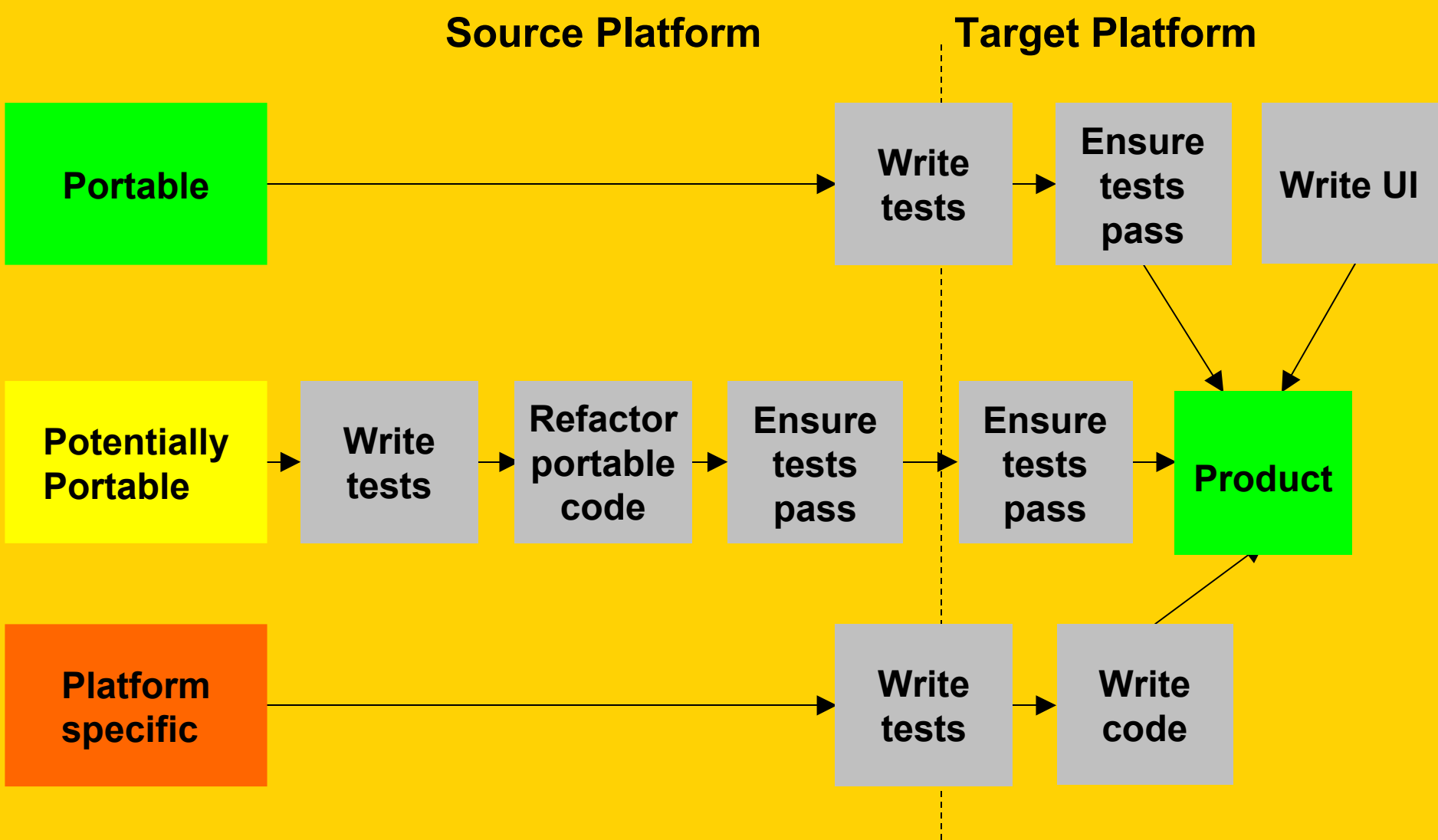
5. Add UI, Networking…

Use existing tests for networking etc.

Interactive (agile?), or specification for UI



Summary: Test-driven porting



Transliteration

Same technique:

- Write tests on source platform
- Transliterate tests to target
- Transliterate code to get tests running

Useful for:

- Porting between different languages
- Code where every line changes…

We’re doing a large C# to Symbian OS port currently

- Source platform tests only initially…

Transliteration Example

```
public class PostalAddressResolution : ISupportInquiries  {
// …
protected string ConstructP1()       {
    string P1 = "";
    if (origItem.UseCityState || origItem.PostalCode.Length == 0)         {
        bool InternationalFormat =
            !(origItem.Country == "US" || origItem.Country == "CN");
        if (InternationalFormat)
            P1 = string.Format("{0},{1}", origItem.City, origItem.Country);
        else
            P1 = string.Format("{0},{1},{2}", origItem.City, origItem.State,
                                        origItem.Country);
    }
    else {
         P1 = string.Format("{0},{1}", origItem.PostalCode, origItem.Country);
    }
    return P1;
}
```

In Symbian OS…

```
class CPostalAddressResolution : public CSupportInquiries
...

HBufC* CPostalAddressResolution ::ConstructP1LC() {
    RPString P1;
    P1.CleanupClosePushL();
    if (iOrigItem->UseCityState() || iOrigItem->PostalCode().Length() == 0) {
        TBool internationalFormat = !(iOrigItem->Country().Compare(_L("US")) == 0
                                    || iOrigItem->Country().Compare(_L("CN")) == 0);
        if (internationalFormat) {
            P1.AppendL(iOrigItem->City());   P1.AppendL(_L(","));   P1.AppendL(iOrigItem->Country());
        } else {
            P1.AppendL(iOrigItem->City());  P1.AppendL(_L(","));    P1.AppendL(iOrigItem->State());
            P1.AppendL(_L(","));  P1.AppendL(iOrigItem->Country());
        }
     } else {
        P1.AppendL(iOrigItem->PostalCode());   P1.AppendL(_L(","));
        P1.AppendL(iOrigItem->Country());
     }

    HBufC* buf = P1.AllocL();    CleanupStack::PopAndDestroy(); // P1
    CleanupStack::PushL(buf);
    return buf;
}
```

Architectural Issues

Threading

- Expensive to port
- Shared writing to UI?

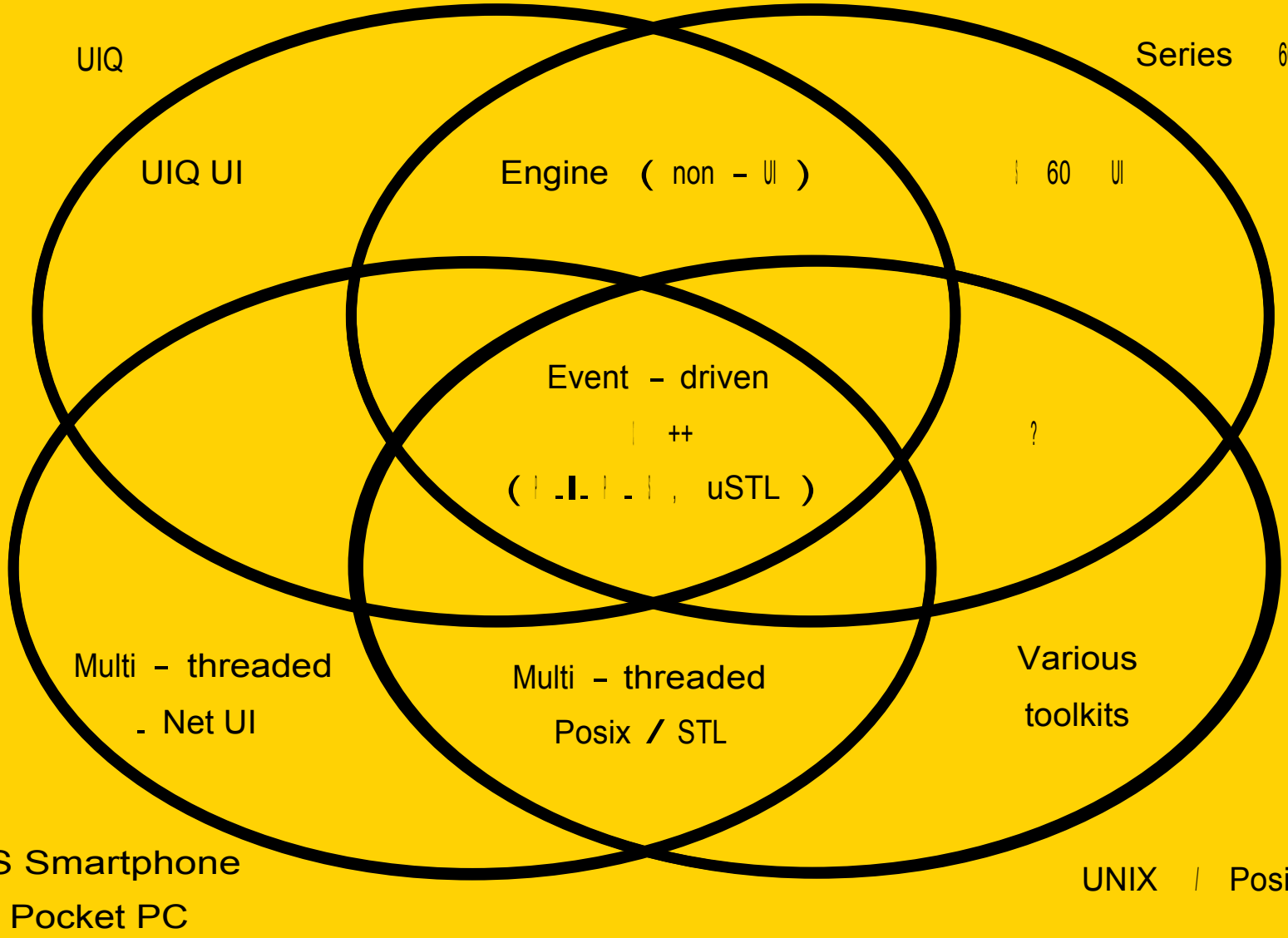
Network

- Posix OK (ish)
- Beware event model

Static data

- No longer really an issue…

Portable Architecture?



Two worst mistakes

We don’t need all that!

- Usually we do…

The big bang blind port

- Need some kind of test (pref. automatic) for each step
- Difficult to get started…

Resources

Downloads

- P.I.P.S, uSTL, etc.

Forum Nokia Tech Lib
UIQ Developer Forum

UIQ 3 Porting Paper,

- Penrillian, SonyEricsson

Links page:

www.penrillian.com/porting

Questions?