Symbian OS C++ - a first look under the hood

John Pagonis

Senior Software Engineer / Senior Developer Consultant

Developer Consulting

symbian

Welcome :-)

- This talk is about the Symbian OS C++ framework
- For people interested in knowing the “Whys”
  - . . . behind some Symbian OS idioms
  - . . . behind Symbian OS C++ dialect and paradigms
  - . . . mobile development
- For people that like to “hack” below the surface
- For people familiar with (Symbian) OS, frameworks, language architecture.
- Why things are the way they are…..
- “Interrupts, help unblocking”

symbian

Copyright © 2006 Symbian Software Ltd.

Why this talk

- Because Symbian OS is different
- Because people may be tempted not to spend time to appreciate it :-/
- To tell people beforehand …
- Because of it's mass adoption and deployment we run a risk of badly educated engineering…
- "Never criticise what you don't understand"

symbian

Copyright © 2006 Symbian Software Ltd.

About architecture…

- Every architectural feat exists within some context, where it evolves under (competing many times) forces. That context and such forces present the architect with constraints and affordances.
- For us ‘’small mobile computers’’ has always been that context.
- We’ve been operating in this context for the past 20+ years or so !
- Like any building that has stood there through time, forces around it may or will change. Fortunately for us software is a bit more adaptable and malleable…

symbian

Copyright © 2006 Symbian Software Ltd.

Some historical background

- Early 80s: Spectrum ZX/81 Flight Simulator and other games
- 80s : 8bit Organiser circa '84 (I, II, P350, CM, LZ64 etc) and the Sinclair QL app suite – early frameworks
- Late 80s : SIBO/Epoc16 circa '88 – WIMP(MC), HWIM(s3), XWIM(s3a) - emerging frameworks
- 90s : Epoc32 – EKA1, HCIL, Eikon – frameworks
- Late 90s – today: Symbian OS – EKA1/EKA2, Uikon, UIQ, S60, MOAP(FOMA) UI etc

symbian

Copyright © 2006 Symbian Software Ltd.

Evolution…

- Games (efficiency and hardware insight)
- Vertically integrated products (hardware and software) and app development
- Product diversification, Frameworks, OS creation and evolution (many of)
- Software reuse and platformisation

- Z80 assembler
- C, systems thinking, UI, OPL, first object-based designs with C, limited comms, early OOP
- OOP/OOD, C++, Java, OPL, comms
- J2ME, Python, Ruby, C++, many GUI frameworks, lots of comms, open to anything and anyone really…

symbian

Copyright © 2006 Symbian Software Ltd.

Symbian OS framework context

- User Interaction and Responsiveness
- Battery Powered
- ‘’Always-on’’ , always pocket-able
- Reliability
- Resource constrained
- Openness…

symbian

Copyright © 2006 Symbian Software Ltd.

Some paradigms…

In order to support the user-centric mentality and operate within the mentioned forces we used some design paradigms and idioms such as :

- Multithreading and pre-emptive multitasking
- Lightweight micro-kernel OS design
- Client-Server, session based IPC among others
- Asynchronous services, Active Objects
- Cleanup Stack, Leaves, Traps for exception handling
- Re-usable frameworks for apps, middleware, GUIs
- …and descriptors

symbian

Copyright © 2006 Symbian Software Ltd.

Language selection… C++

- C++ is a multi-paradigm programming language
- Probably not the best language for loosely coupled open world systems
- But we pretty much knew what the frameworks and apps would need and should be like
- It supports OOP
- Good for low level OS construction, strongly typed and works nicely with assembler.
- Has too many features and demands proper attention, only got standardised recently though…

symbian

Copyright © 2006 Symbian Software Ltd.

Standard C++ ???

Domain forces and late standardisation meant :

- We had to came up with our own exception mechanisms
- Came up with our own collections, container and buffer/string handling frameworks
- Concurrency support was not in the package

- …and we have been criticised ever since for designing something different that works and is domain specific :-(

symbian

Copyright © 2006 Symbian Software Ltd.

Symbian OS C++

We refer to Symbian OS C++ as the C++ domain specific dialect and accompanying frameworks that we use to build Symbian OS and the software that runs on it.

- This dialect as well as the OS itself have emerged from the vast experience dealing with small mobile computers.
- This experienced has been captured in the domain specific idioms and paradigms of Symbian OS.
- **Some of the idioms and paradigms present in Symbian OS are reflected in Symbian OS C++ and vice versa.**

symbian

Copyright © 2006 Symbian Software Ltd.

Microkernel design

- Lightweight Microkernel with client server architecture
- System servers as well as user servers run in user space
- Kernel uses Virtual Memory Model and MMU for memory protection
- Driver code runs kernel space
- Client/server session based IPC for server access
- Servers mediate access to shared resources and services

symbian

Copyright © 2006 Symbian Software Ltd.

EKA2

- Multi-threaded, pre-emptible Real Time Kernel
- In-fact is a Symbian OS personality on top of a Nanokernel – **many** personalities can co-exist ;-)
- O(1) scheduler
- System calls are all pre-emptible as well → dual stacks
- Deterministic ISR, thread response, latencies etc
- Memory models and local allocator strategies can be plugged-in
- Many more IPC/ITC mechanisms such as local/global message queues, publish-subscribe, global anonymous queues, shared and global memory chuncks

symbian

Copyright © 2006 Symbian Software Ltd.

Kernel – Nano Kernel and personalities

- EKA2 splits the kernel in two layers
- A Nano Kernel and a Symbian OS Kernel
- The Symbian OS Kernel is still a Micro Kernel
- The Symbian OS Kernel is a ‘’personality’’ on top of the NK
- There can be many such personalities simultaneously running, thus many kernels can be run pre-emptively on top of the Nano Kernel !!!!!!
  - For example, one for the GSM or UMTS stack and one for Symbian OS
- NK is responsible for the very basic synchronisation, timing, initial interrupt handling and scheduling services
- It is not depended on the system library (euser) and doesn’t know about processes or memory models
- All offered services are deterministic with bound execution times

Copyright © 2006 Symbian Software Ltd.

Kernel pre-emptibility

- The EKA2 Symbian OS Kernel is multi-threaded
  - … Device drivers are much easier to write
- It is completely pre-emptible
  - … even the memory allocations and context switch can be pre-empted
- User side threads have a user mode and supervisor mode stack
  - … executive calls run on user thread's supervisor stack
  - … executive calls can thus all be pre-empted !!!!

symbian

Copyright © 2006 Symbian Software Ltd.

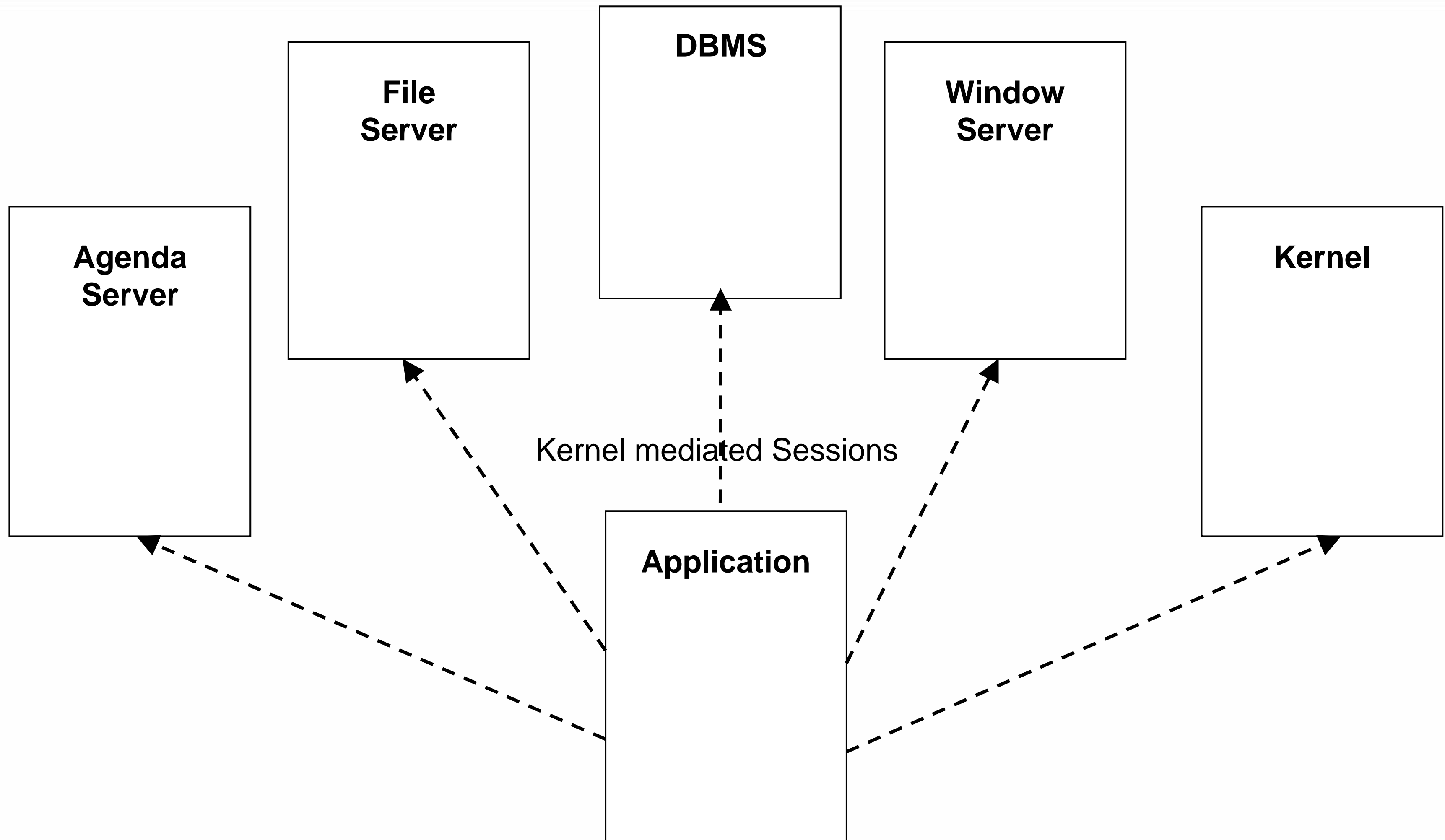
Three Symbian OS C++ paradigms

- Active Objects
- Exceptions handling
- Descriptors and run time bounds checking

symbian

Copyright © 2006 Symbian Software Ltd.

Asynchronous services



symbian

Copyright © 2006 Symbian Software Ltd.

Many asynchronous services….

- But the OOP model is **serial** and so is vanilla C++
- Concurrency support was needed for asynchronous event handling and the multitude of client-server comms
- A lightweight model that introduces concurrency and **avoids synchronisation issues** is that of Active Objects
- btw: Grady Booch talks about AOs in his 1990 book on ‘’Object Oriented Design with Applications’’. Defined as objects having their own thread of control.
- We used collaborative AOs within a thread instead

symbian

Copyright © 2006 Symbian Software Ltd.

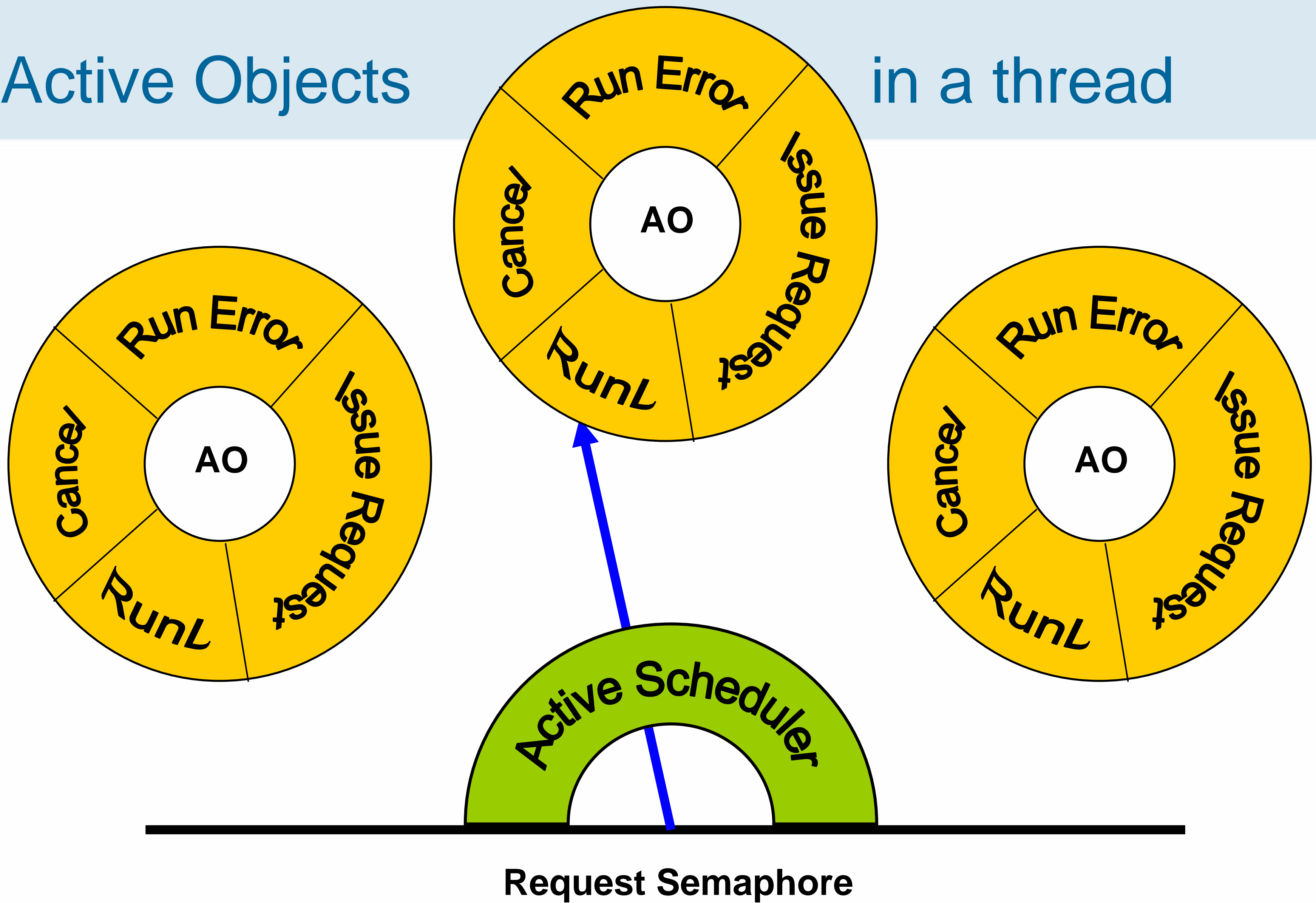
Active Objects

- Thus every thread got a *request semaphore*
- Most threads and certainly all apps and servers got an *Active Scheduler*
- So that it can schedule…. Active Objects
- Thus Active Objects became a lightweight mechanism to **encapsulate concurrent transactions** over session based IPC
- Simple: Issue a request, run when the request completes, if an error occurs handle the error.
- … priorities, run to completion, no pre-emption

symbian

Copyright © 2006 Symbian Software Ltd.

Active Objects in a thread



Copyright © 2006 Symbian Software Ltd.

symbian

The Active Objects interface

```
class CActive : public CBase
    {
public:
    IMPORT_C ~CActive();
    IMPORT_C void Cancel();
    IMPORT_C void Deque();
    IMPORT_C void SetPriority(TInt aPriority);
    inline TBool IsActive() const;
    inline TBool IsAdded() const;
    inline TInt Priority() const;
protected:
    IMPORT_C CActive(TInt aPriority);
    IMPORT_C void SetActive();
    virtual void DoCancel() =0;
    virtual void RunL() =0;
    IMPORT_C virtual TInt RunError(TInt aError);
public:
   TRequestStatus iStatus;
    }
```

symbian

Copyright © 2006 Symbian Software Ltd.

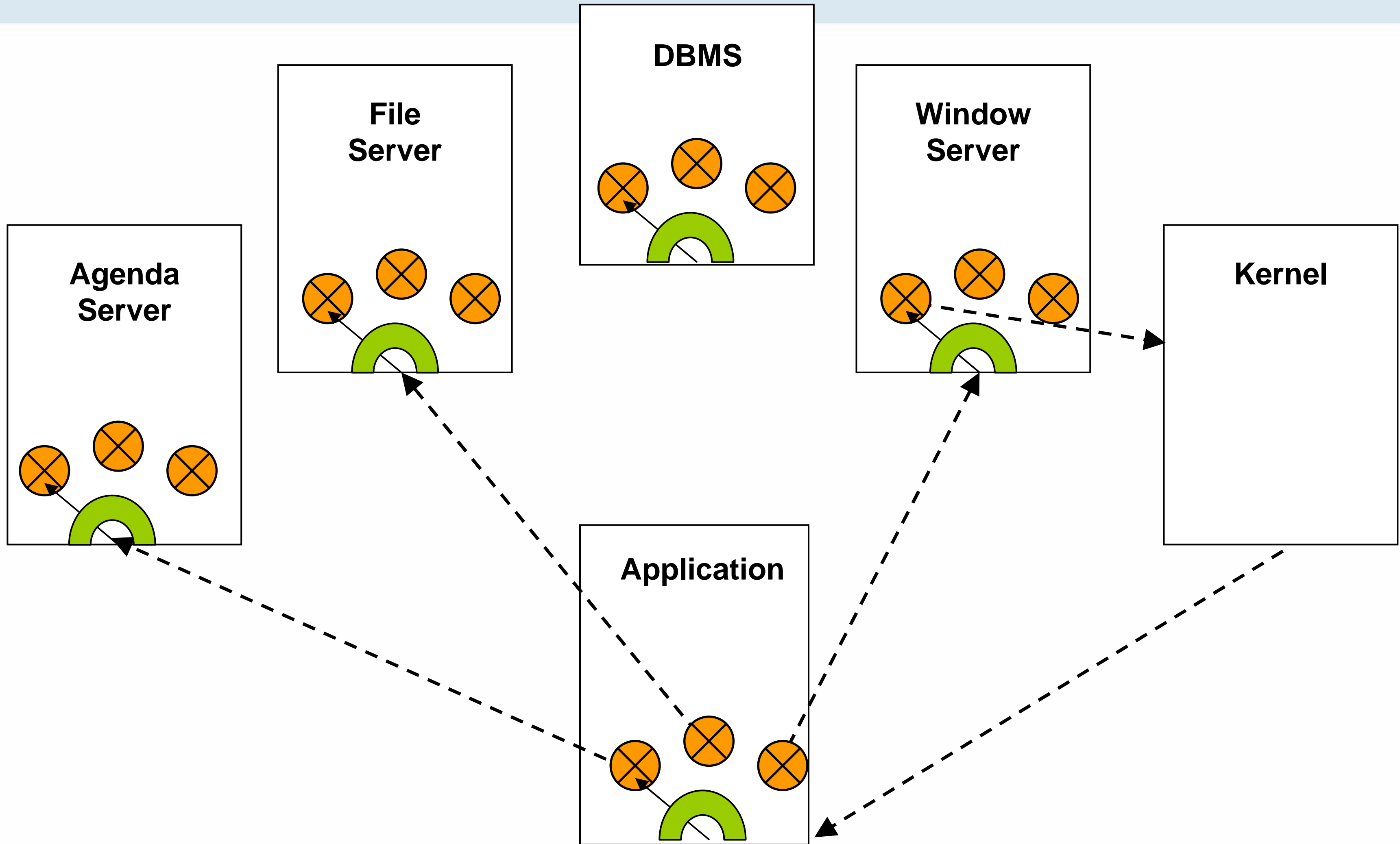
Active Object transaction encapsulation

- Use or open a session to some service provider (server)
- Add AO to the Active Scheduler
- Issue a request to that service that may complete asynchronously
- Set the AO active, thus notify the scheduler
- … handle the completion of the request in the `RunL`
- If there is an unhandled exception `RunError`
- ..or let the Active Scheduler know

Copyright © 2006 Symbian Software Ltd.

symbian

Active Objects almost everywhere



Copyright © 2006 Symbian Software Ltd.

Good things about Active Objects

- AOs help to manage complexity and synchronisation issues with concurrency within a thread without too much to worry about
- AOs encapsulate transactions and their error handling
- AOs help us avoid multi-threading for async event handling where it is not always needed or when it is more complicated or just an overkill
- AOs let servers manage many clients and many transactions with just one thread

symbian

Copyright © 2006 Symbian Software Ltd.

..and the bad

- BUT they are **no panacea** and should not be used where they don't fit the paradigm !!!
- Multi-threading instead of AOs is valid and must be used where it must !

Copyright © 2006 Symbian Software Ltd.

symbian

Symbian OS C++ exceptions handling

Native Symbian OS C++ programs use an exceptions handling mechanism which is not compatible with the ISO C++ 0x exceptions handling way (actually the design of Symbian OS predates ISO/IEC 14882 1998/2003).

In Symbian OS, programmers have to think and implement their trapping, throwing and heap allocated object tracking in the face of exceptions, manually - as part of their design and not as an afterthought.

Symbian OS C++ exceptions handling philosophy, is to be ‘’in your face’’, but not ‘’in the way’’.

symbian

Copyright © 2006 Symbian Software Ltd.

About exception handling…

```
SomeMethodL()
    {
    tmpObject = new (ELeave) CHeapyObject()
    CleanupStack::PushL(tmpObject)
    DoSomethingThatMayLeaveL()
    CleanupStack::Pop(tmpObject);
    // …and pass ownership of that object
    //OR possibly
    CleanupStack::PopAndDestroy(tmpObject)
    }
```


Copyright © 2006 Symbian Software Ltd.

And why is that ?

- Poor or no support for C++ exception handling in compilers at the time
- CleanupStack is in your face → a good thing
- Compilers behind the scenes write all the exception handling code for you, they tend to be really conservative and churn much more code than a developer would
- ….this may change with newer compilers

symbian

Copyright © 2006 Symbian Software Ltd.

How does the CleanupStack work ?

- It stores pointers to objects to be destroyed in case of an exception (a.k.a. Leave)
- These pointers are stored in nested levels
- Such levels are marked by the TRAP macro (think _almost_ of ‘C’ setjump/longjump here with exec call)
- When a Leave occurs, it makes sure it calls all d’tors (and cleanup items) of objects belonging to the corresponding Trap level.
- And the stack unwinds to the point of TRAP, returning an exception error code.

symbian

Copyright © 2006 Symbian Software Ltd.

Trapping and Leaving

```
Cleanupstack::PushL(something);/* will stay on
   the CleanupStack if it Leaves below */

TRAPD(err,SomeMethodL());

if (KErrNone==err)

   {//do something }

else

   {//start handling the exception

   //and possibly propagate it by another  //Leave

   }

….possibly pop something
```

Copyright © 2006 Symbian Software Ltd.

symbian

(Almost) Introducing Modern ISO Standard C++…

- TRAP and User::Leave() are now implemented internally in terms of catch and throw
- Ported 3rd party code can use standard C++ exception mechanisms

  → try/catch/throw

  … But not mix these with Symbian OS system APIs

  … “Leaving functions must not throw, unless they also catch internally” → “barriers”
- Standard C++ exception specifications are supported
- Writeable static data for DLLs is finally here !!

  … emulator has a caveat that allows only one DLL attachment though

symbian

Copyright © 2006 Symbian Software Ltd.

..more

- The C++ spec is ISO/IEC 14882 1998/2003 and is enabled but not delivered.
- RTTI and dynamic_cast<> is enabled, but not for Symbian OS APIs.
- RTTI - dynamic_cast<> implies the use of catch() to handle the exception that is thrown when the cast fails, and this doesn't mesh well with the limited use of C++ exceptions for TRAP and Leave().
- Current coding standards are to use NONSHARABLE_CLASS for internal component classes, which disables the RTTI info being emitted.

symbian

Copyright © 2006 Symbian Software Ltd.

…therefore remember

- Symbian OS is now compiled with exceptions turned on, i.e. exception tables and the like are included in the (x86 and EABI) binaries
- implemented User::Leave/TRAP in terms of exceptions
- It is possible to throw and catch exceptions within your own code.
- It is possible to catch leaves and exceptions in your own code….Don't
- You cannot, in general, TRAP an exception
- Code that leaves and code that throws should not intermingle
- nor should code that leaves have objects on the stack with non-trivial destructors
- nor should code that throws, use the cleanup stack OR depend on code that indirectly depends on objects on it.
- Code that leaves MAY NOT call code that throws, without some suitable barrier -> will not call the cleanup sequence

Copyright © 2006 Symbian Software Ltd.

symbian

On Barriers

- Q: Why do we need a 'suitable' barrier?
- A: To protect the integrity of the cleanup stack.

```
#ifdef __LEAVE_EQUALS_THROW__
EXPORT_C void User::Leave(TInt aReason)
    {
    TTrapHandler* pH = Exec::TrapHandler();
    if (pH)
         pH->Leave(aReason);// causes things on the cleanup stack to be
    cleaned up
    throw XLeaveException(aReason);
    }
#endif
```

Copyright © 2006 Symbian Software Ltd.

symbian

A C++ teaser…. be prepared !

- Spot any problems ?

```
void SymbianFunc1L()
    {
    Cx* obj1 = Cx::NewLC();
    NonSymbianFuncL(obj1);
    CleanupStack::PopAndDestroy(obj1);
    }
void NonSymbianFuncL(Cx* aCx)
    {
    Sx obj2(aCx); // automatic object with destructor
    SymbianFunc2L(obj2.DoSomething());
    obj2.DoSomethingElse();
    // obj2 destroyed here
    }
void SymbianFunc2L( aParam )
    {User::Leave( leavecode );}
```

symbian

Copyright © 2006 Symbian Software Ltd.

Descriptors not strings!

- Symbian OS C++ makes use of objects that encapsulate buffers or pointers to buffers as opposed to naked ‘C’ strings
- Descriptors are everywhere
- Descriptors **do run-time bounds checking**
- Descriptors are ROMable
- Descriptors can describe buffers of ASCII, Unicode and binary data on the heap or on the stack
- They offer a safe and consistent mechanism for dealing not only with strings but also with binary data

symbian

Copyright © 2006 Symbian Software Ltd.

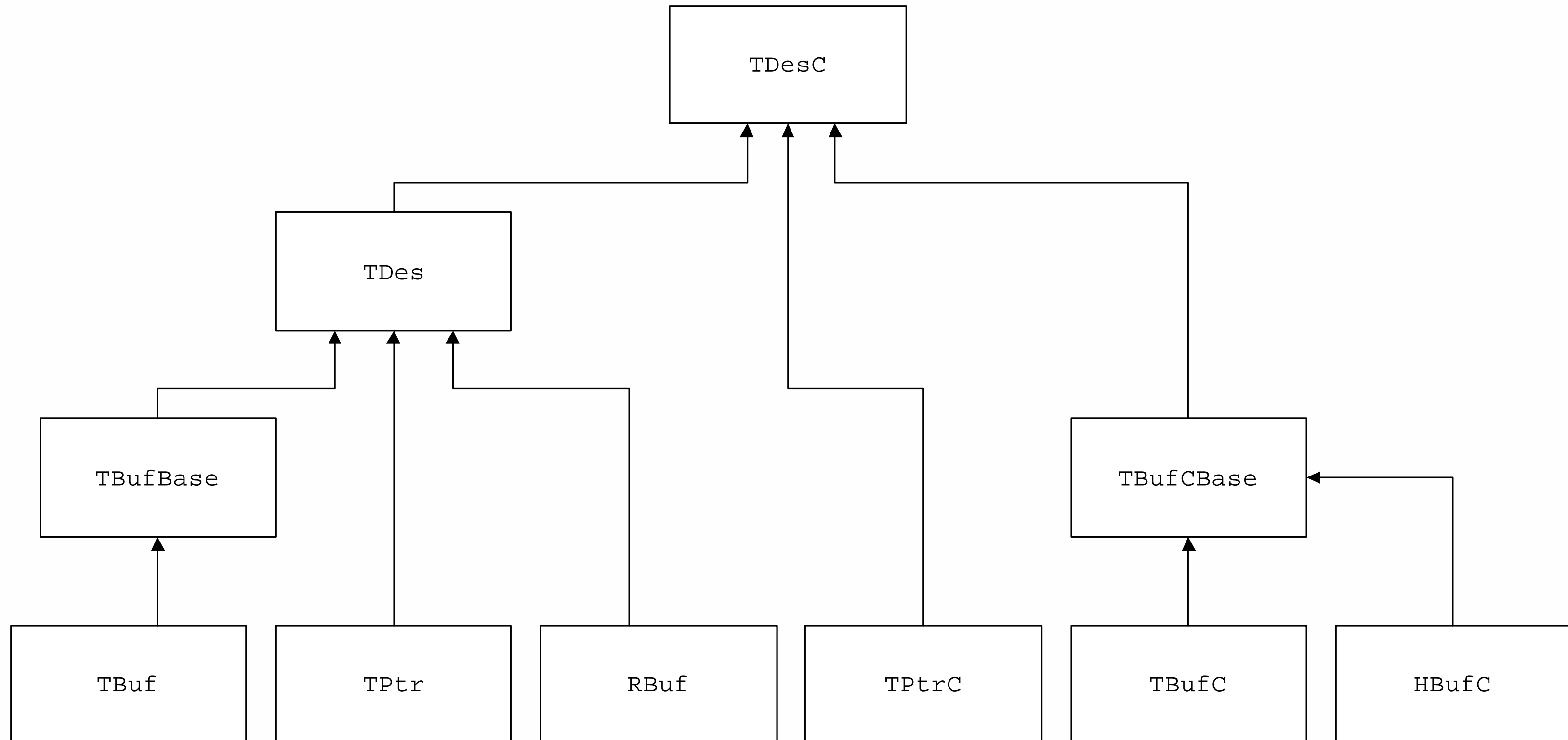
Descriptor classes

- All descriptor classes inherit from the base class `TDesC`
- There are four 'abstract' classes: `TDesC`, `TDes`, `TBufBase` and `TBufCBase` (TBufCBase and TBufBase are an implementation convenience)
- These are not abstract in the C++ sense as descriptors do not have pure virtual methods; however they are abstract in the sense that they are not intended to be instantiated.
- There are six concrete descriptor types: `TBuf`, `TPtr`, `RBuf`, `TPtrC`, `TBufC` and `HBufC`
- In addition there is a related type known as a 'literal'.'Lits' are not really descriptors but they are closely related.
-

symbian

Copyright © 2006 Symbian Software Ltd.

Descriptors hierarchy



Copyright © 2006 Symbian Software Ltd.

Descriptor attributes

Type	Constness	Name	Approximate C equivalent
Literal	**Not modifiable**	TLitC	static const char[]
Stack Descriptor	**Modifiable**	TBuf	char[]
Stack Descriptor	**Not directly modifiable**	TBufC	const char[]
Pointer Descriptor	**Modifiable**	TPtr	char* (pointing to non-owned data)
Pointer Descriptor	**Not modifiable**	TPtrC	const char* (pointing to non-owned data)
Heap Descriptor	**Modifiable**	RBuf	char* (pointing to owned data
Heap Descriptor	**Not directly modifiable**	HBufC	const char* (pointing to owned data)

Copyright © 2006 Symbian Software Ltd.

symbian

Descriptor memory layout (for the curious:-)

TDesC	Type	Length		
TDes	Type	Length	Max length	
TLitC	Type (0)	Length	Data	
TBufC	Type (0)	Length	Data	
HBufC	Type (0)	Length	Data	
TPtrC	Type (1)	Length	Pointer to data	
TBuf	Type (3)	Length	Max length	Data
TPtr	Type (2 or 4)	Length	Max length	Pointer to data or pointer to descriptor
RBuf	Type (2 or 4)	Length	Max length	Pointer to data or pointer to HBufC

Copyright © 2006 Symbian Software Ltd.

symbian

Basic usage

```
// Create a literal descriptor
_LIT(KTxtHelloWorld, "Hello World!"); //TLitC
const TInt KHelloWorldLength = 12;
// create a TBuf
TBuf<KHelloWorldLength> tbuf(KTxtHelloWorld);
// create a TBufC
TBufC<KHelloWorldLength> tbufc(KTxtHelloWorld);
// create a HBufC
HBufC* hbufc = KTxtHelloWorld().AllocLC();
// Create an RBuf
RBuf rbuf;
rbuf.CreateL(KHelloWorldLength, KTxtHelloWorld);
rbuf.CleanupClosePushL();
// create a TPtrC
TPtrC tptrc(tbufc);
// create a TPtr
TPtr tptr = *hbufc;
```

symbian

Copyright © 2006 Symbian Software Ltd.

About bounds checking

- You can't get buffer overflows with descriptor APIs!!
- Append, At, [], Format and Copy will panic your thread before you overwrite any data!
- Makes you fix it really quickly as opposed silently tripping over :-)
- ASCII to Unicode conversion comes for free with Copy()

…and much much more

symbian

Copyright © 2006 Symbian Software Ltd.

Thank you !

Q & A

For more:

"Symbian OS Explained", by Jo Stichbury

"Symbian OS for Mobile Phones vol2" , by R. Harrison

"Symbian OS Internals: Real Time Kernel Programming", Jane Sales et al

..and of course visit the Symbian Developer Network

symbian

Copyright © 2006 Symbian Software Ltd.