Proposals 4 parts

Virtual Accelerator
Open CBCM
Data Base Cycle Server
Quick Fix

About these proposals

- I am trying to present things in a logical order, not chronologically or in order of importance.
- This means I need to deal with ambiguities and definitions first.
- I am looking for simplifications

About Event payloads and telegrams

- Each event sent over the timing cable is able to piggyback a 16 bit payload.
- This payload can be used to pilot multiplexing in RT tasks and timing receiver hardware modules.
  - It can also be used for other things, beam energy for example.
- Today in the PS the payload is not used, it always contains zero.
- Today in the SPS the payload contains the USER and drives multiplexing.

More about payloads and telegrams

- A telegram is a way of distributing information about the current and next machine cycles.
- The telegram contains parameters such as User, Destination, Particle-Type etc. Each accelerator has its own custom built telegram that contains parameters relevant to that machine.
- The telegram is valid 1ms before the start of the cycle until 1ms before the end of the cycle.

Even more about...

- Telegrams suffer from an interpretation problem when an RT task is connected to a timing event close to the start of the cycle.
- If the event is a forewarning, the RT task must use the NEXT cycle description.
- If the event occurs after the start cycle the PRESENT must be used.
- Great care must be taken when moving events around.
- Payloads do not suffer from this problem because the multiplex parameter is carried in the event itself.

Finally on payloads and telegrams

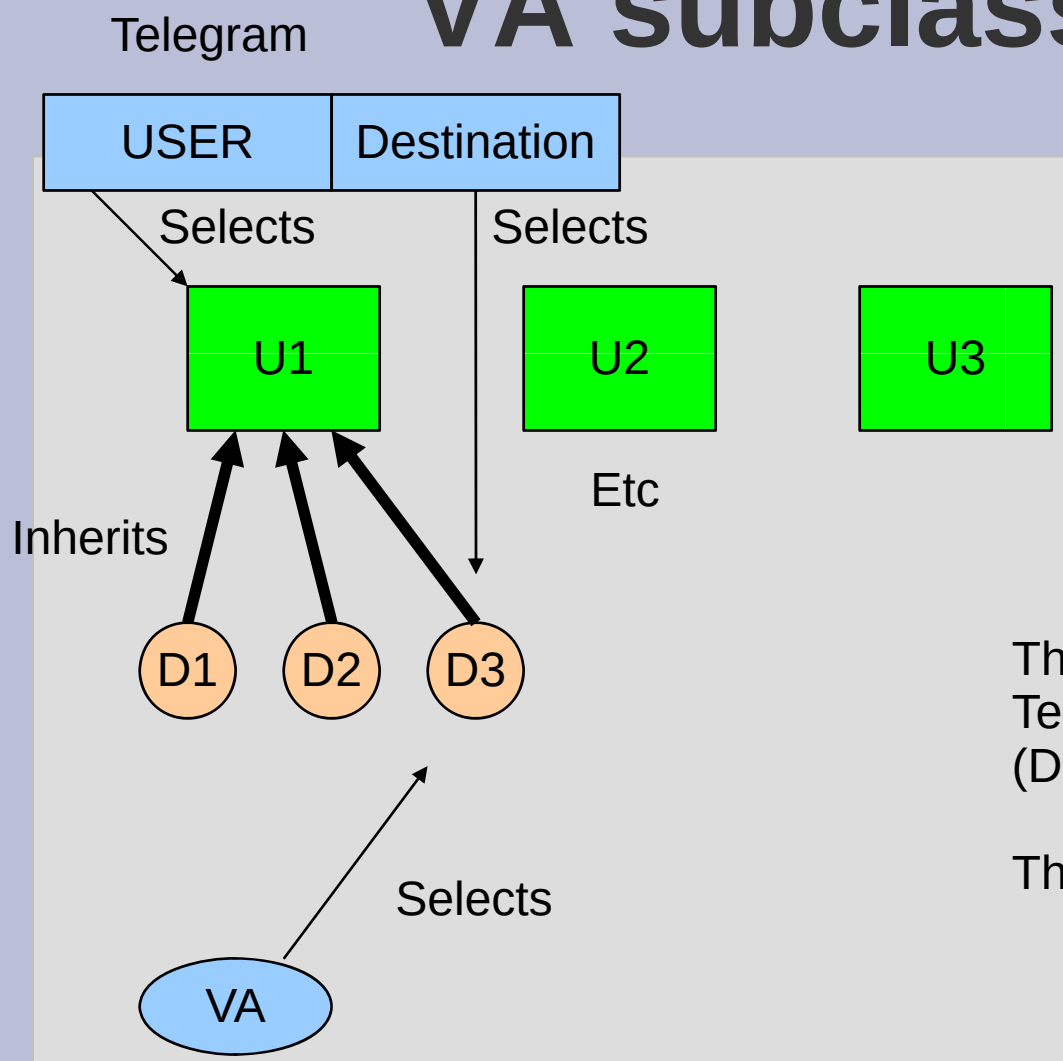
- The payload in the CPS complex timing is always zero, multiplexing is done by using the USER parameter from the telegram.
- Using other parameters in the telegram to drive multiplexing can break the archives
  - Double and Triple PPM multiplexes on other parameters without breaking the archives by storing extra data with the user.

- Conclusion
  - Payloads are a better way of driving multiplexing

Virtual Accelerator and the Telegram in front ends

- Because a USER is a collection of VAs, and because we multiplex on USER, the RT tasks in the front ends, and timing receiver modules, need telegrams at run time in order to choose which VA to instantiate from the USER settings.
  - So... if multiplexing was on VA rather than USER, then RT tasks and timing receiver modules would not need telegrams.
    - All telegram libraries, double and triple PPM, REGA could be suppressed.
    - A lot of RT task and hardware logic that decides on which VA to instantiate would no longer be needed.
    - The central timing logic that calculates telegrams could be suppressed.

VA subclasses USER



All can be archived by USER

The VA just addresses directly
Telegram addresses in two stages
(Double PPM!!)

The amount of memory used is the same

What is the CBCM ?

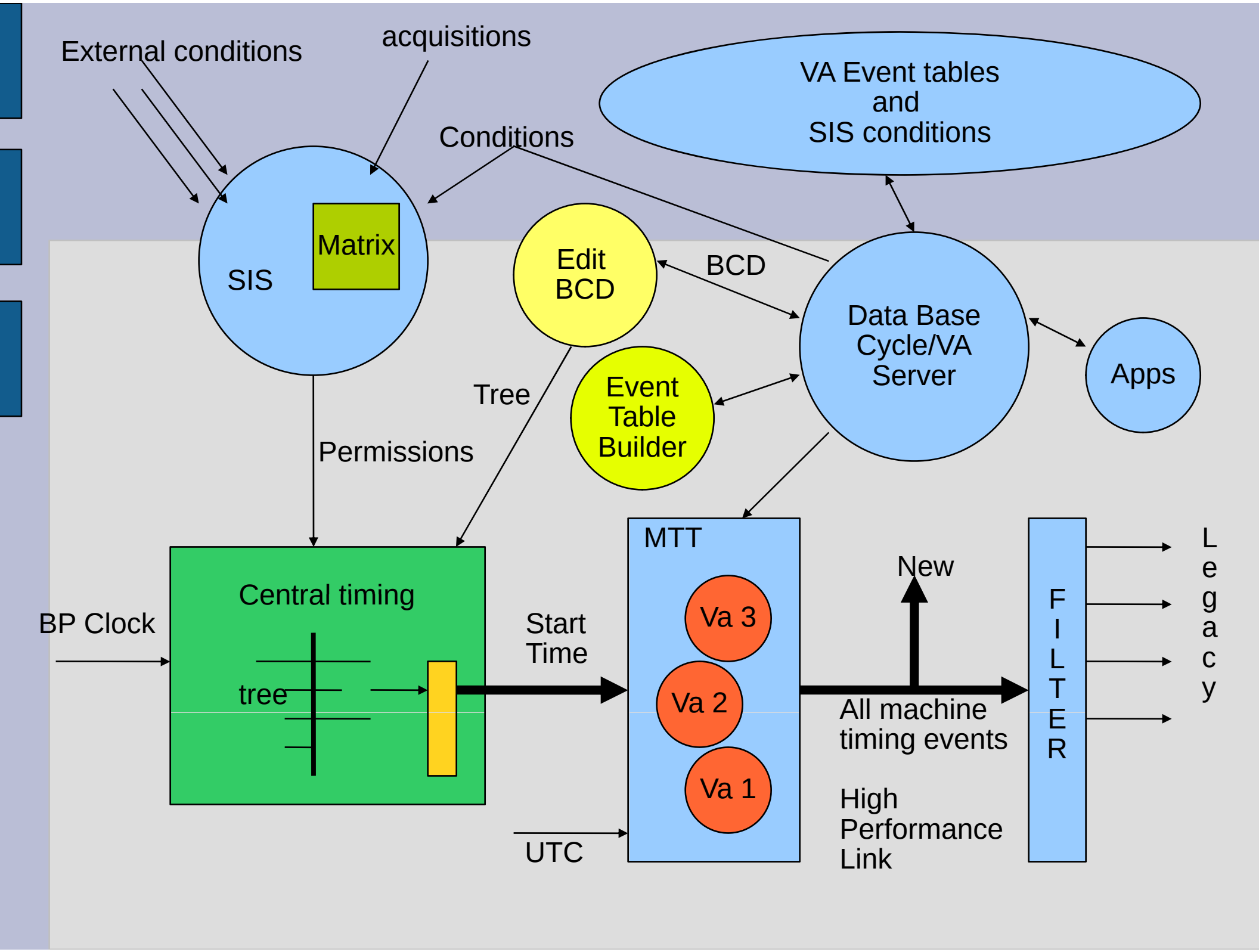
- CERN Central Beam and Cycle Manager
  - It manufactures the timing for the LHC Injector Chain, and the ADE.
  - It builds the telegrams needed by the front end computers to produce settings from USER/s.
  - External conditions logic determines which cycles/beams to execute.
  - Other logic tunes the telegrams.
- It sends corresponding timing events
- FIDO is used to encode all this behaviour
  - FIDO is user hostile, flexible, and not open and transparent. Expert only

Open CBCM a proposal

- Principles
  - It must be transparent, open and simple
  - Use one approach to LHC and LIC timing
    - Build an enhanced version of the MTT card (PCI bus running under Linux).
  - It should be easy and simple to maintain
  - Operations must have full control over its behaviour.
  - It should be able to respond rapidly to external triggers, Dump, Fast Economy Mode Switch

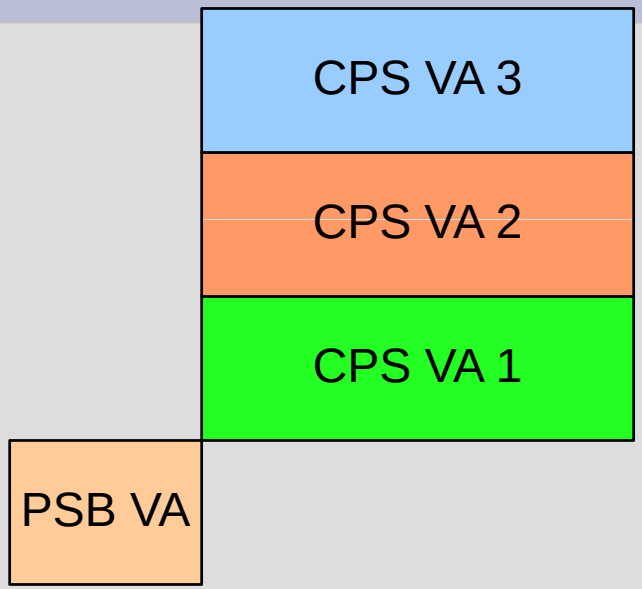
Open CBCM

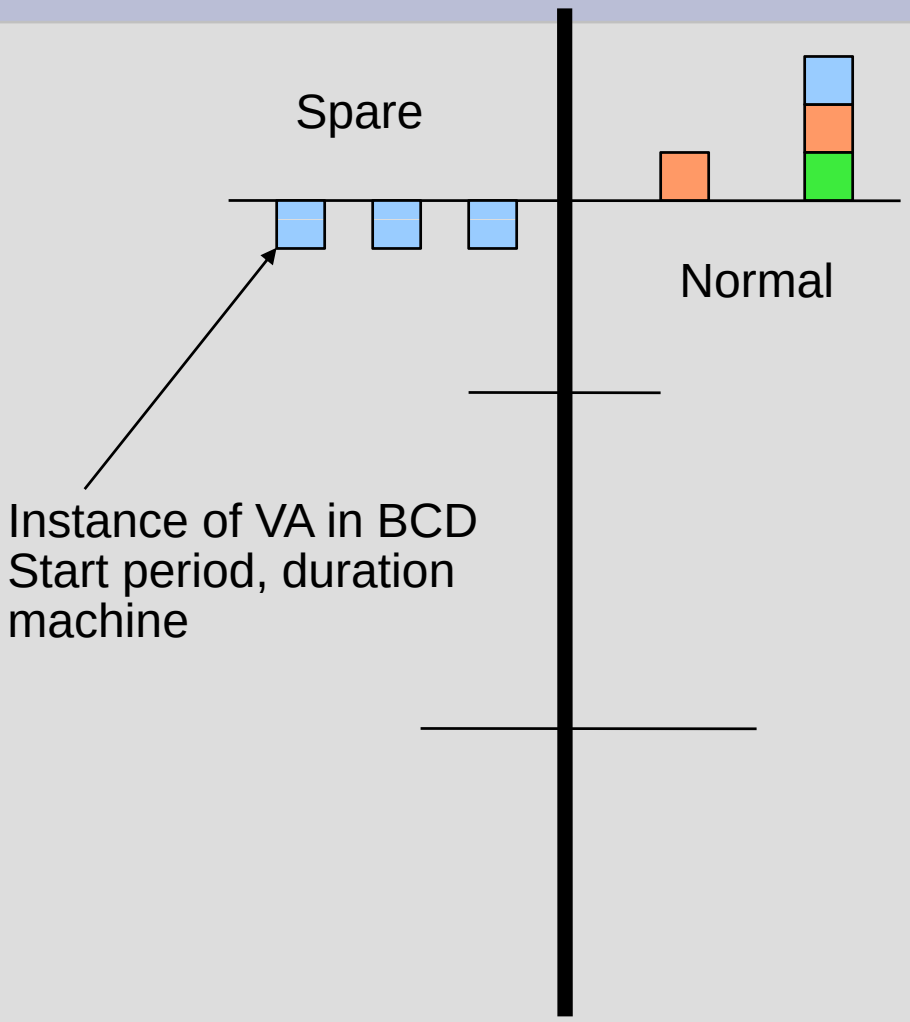
- Open CBCM
  - All FIDO logic is suppressed.
  - SIS controls cycle requests and hence which cycles/beams execute.
  - Cycle variations (Virtual Accelerators) are represented as event tables running on the MTT.
  - Event tables are stored in Oracle/LSA archives in VA archives. No on the fly cycle modifications.
  - Input channel available allows using it to transmit real time data such as Beam Energy, Safe Beam Flags, Post Mortem Switch ...
  - Sending telegrams is still optional


External conditions
acquisitions
VA Event tables
and
SIS conditions
Conditions
SIS
Matrix
Edit
BCD
BCD
Data Base
Cycle/VA
Server
Apps
Event
Table
Builder
Tree
Permissions
MTT
Central timing
BP Clock
tree
Start
Time
Va 3
Va 2
Va 1
New
All machine
timing events
High
Performance
Link
UTC
FILTER
Legacy

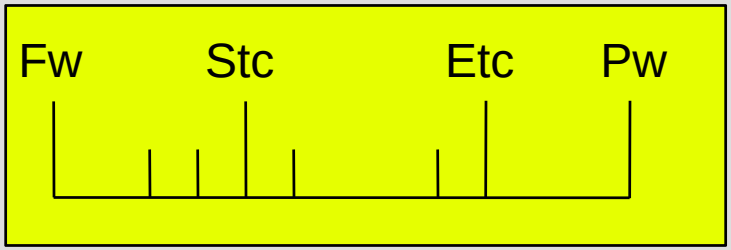
Beam tree

Rende's law: The forewarning of any cycle should be less than the length of the cycle itself
Therefore: Anyone VA can never execute more than twice at the same time
Therefore: We need two virtual processors per accelerator





Event table for a VA



Start = BP + Offset - Fw ⟶ MTT

Open CBCM conclusion

- SIS determines which VAs can be played
  - Standard open operational tool
  - The central timing plays what it can !
- Cycle Variations are VAs loaded as event tables
  - They are just archives
  - Each VA has a set of hardware conditions used by SIS to form the request
- Result
  - Completely open standard system
  - Very simple principles
  - No special FIDO logic
  - One approach for all accelerators
  - Telegrams/VA are still optional for the rest of controls
    - Keep them in the event table

Applications and telegrams

- Why do applications need to be sent telegrams ?
  - Because they need to know what's going on !
- Today telegrams are calculated on the fly by the CBCM so that the front ends can instantiate a VA from the USER data.
- If multiplexing was on VA, then application would not need to be SENT the telegram. They already have them in the archives, all they need to know is which VA is playing by subscribing to an XTIM ...

Applications and VA

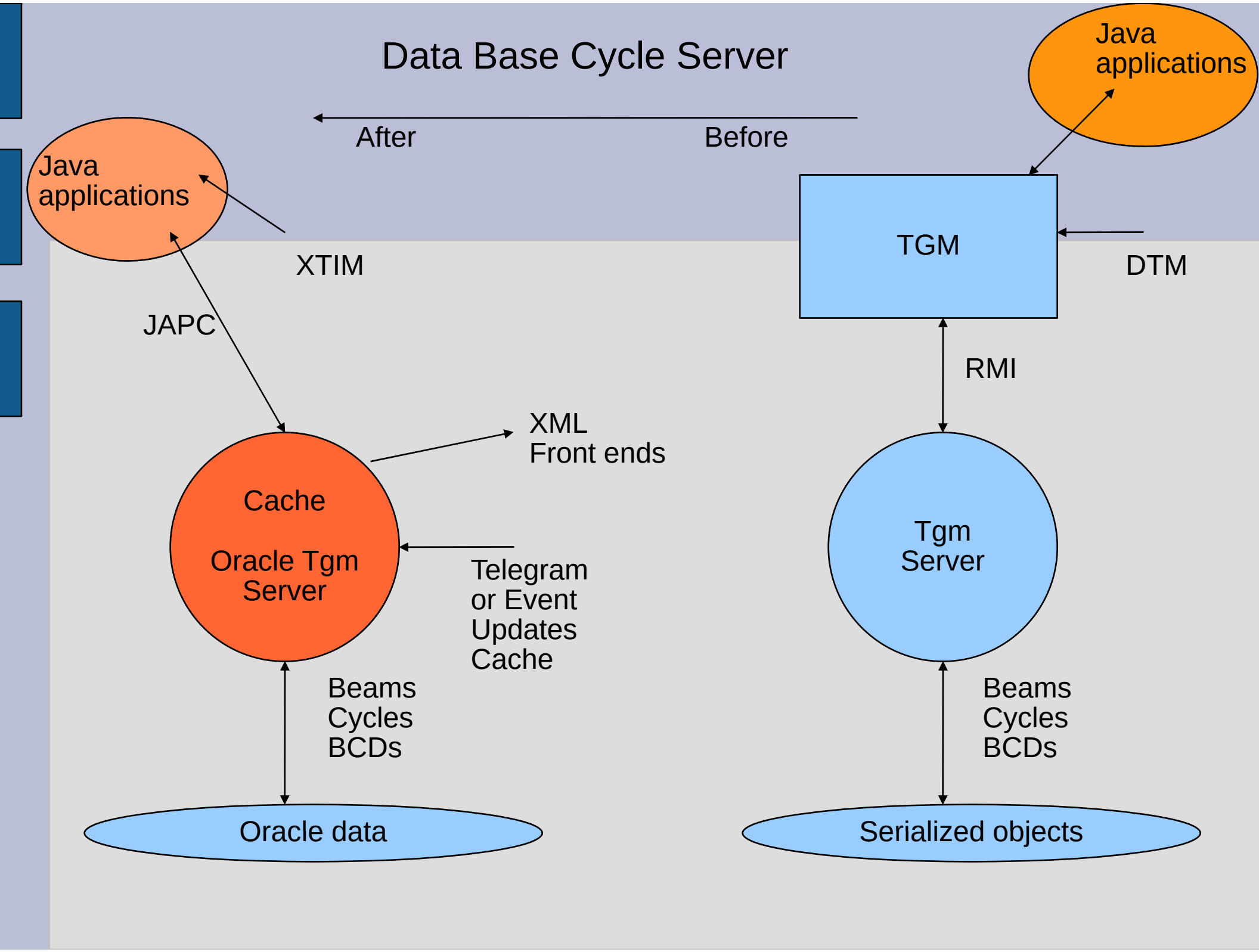
- An application can subscribe to a timing event for synchronization (XTIM).
- When the event arrives it carries the VA ID in its payload.
- It uses this payload to obtain whatever data from the VA archive it needs.
- It can do this by accessing the cache of a data base cycle/VA server
  - Its time to have a way to subscribe to data in Oracle via JAPC/FESA

Virtual Accelerator summing up

- The VA accesses cycle descriptions in the data base cached and maintained at the application level across JAPC.
- We no longer need to send telegrams, but we can if we want.
- The VA is in the event payload. It drives all front-end multiplexing, and application access to run time cycle description for applications.
- All telegram libraries, Java packages and telegram subscriptions are not needed. We can survive with XTIM subscriptions only.
- Many simplifications can be made in the front ends, the central timing and in applications.
- Only standard kit is needed, middleware DB

Data Base Cycle Server

- Replaces most of the Java telegram application package.
- Standard device/property JAPC interface.
- Access to cached cycle descriptions is available at run time.
- Access to Cycles(tgm), Beams and BCDs is available.
- Access could be by User, but VA is better.
- XML file could be produced automatically and accessed in front ends to get some cycle data. (Not the XML file Michel was talking about)

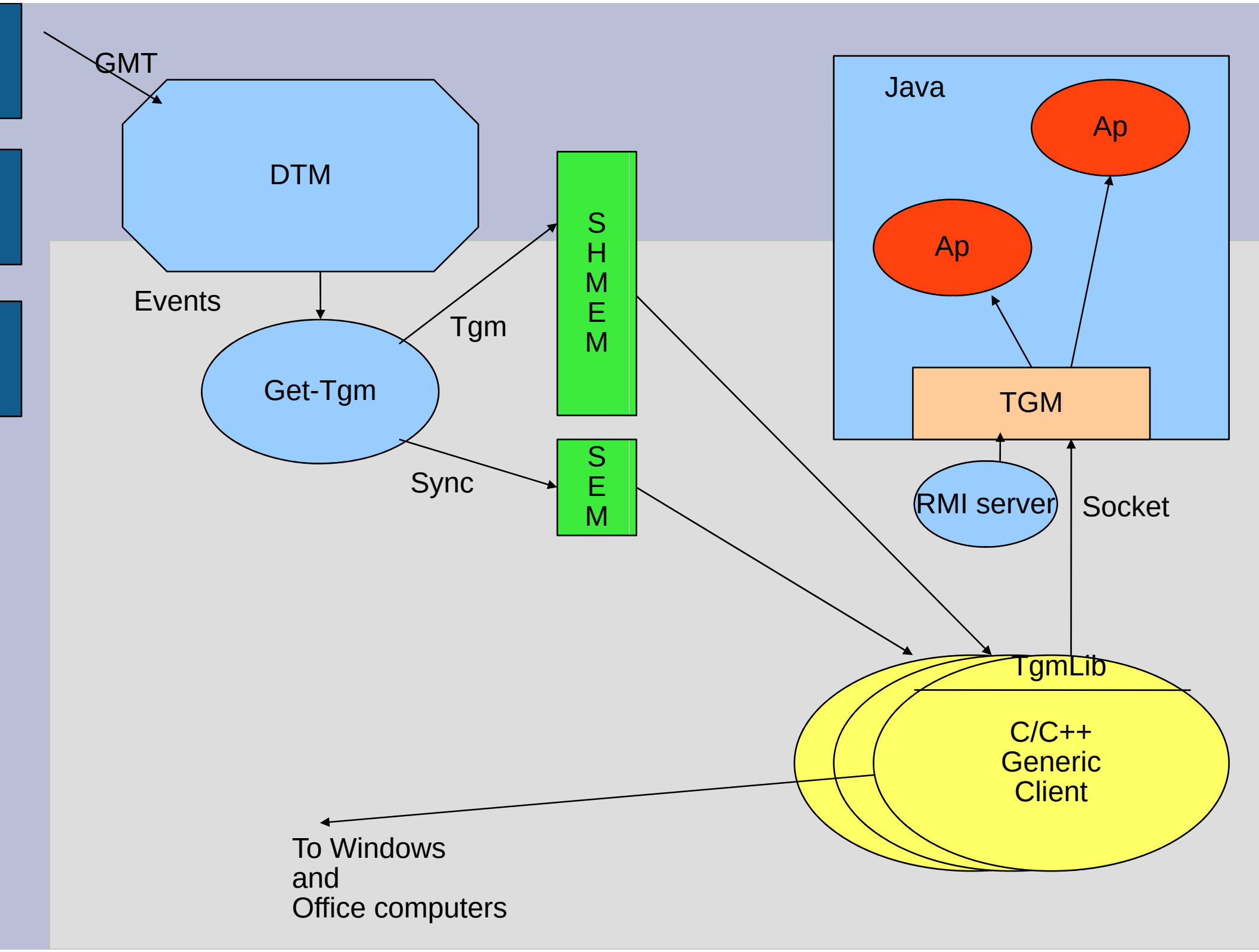

Data Base Cycle Server
Java applications
After
Before
Java applications
XTIM
TGM
DTM
JAPC
RMI
XML
Front ends
Cache
Oracle Tgm Server
Telegram or Event Updates Cache
Tgm Server
Beams
Cycles
BCDs
Beams
Cycles
BCDs
Oracle data
Serialized objects

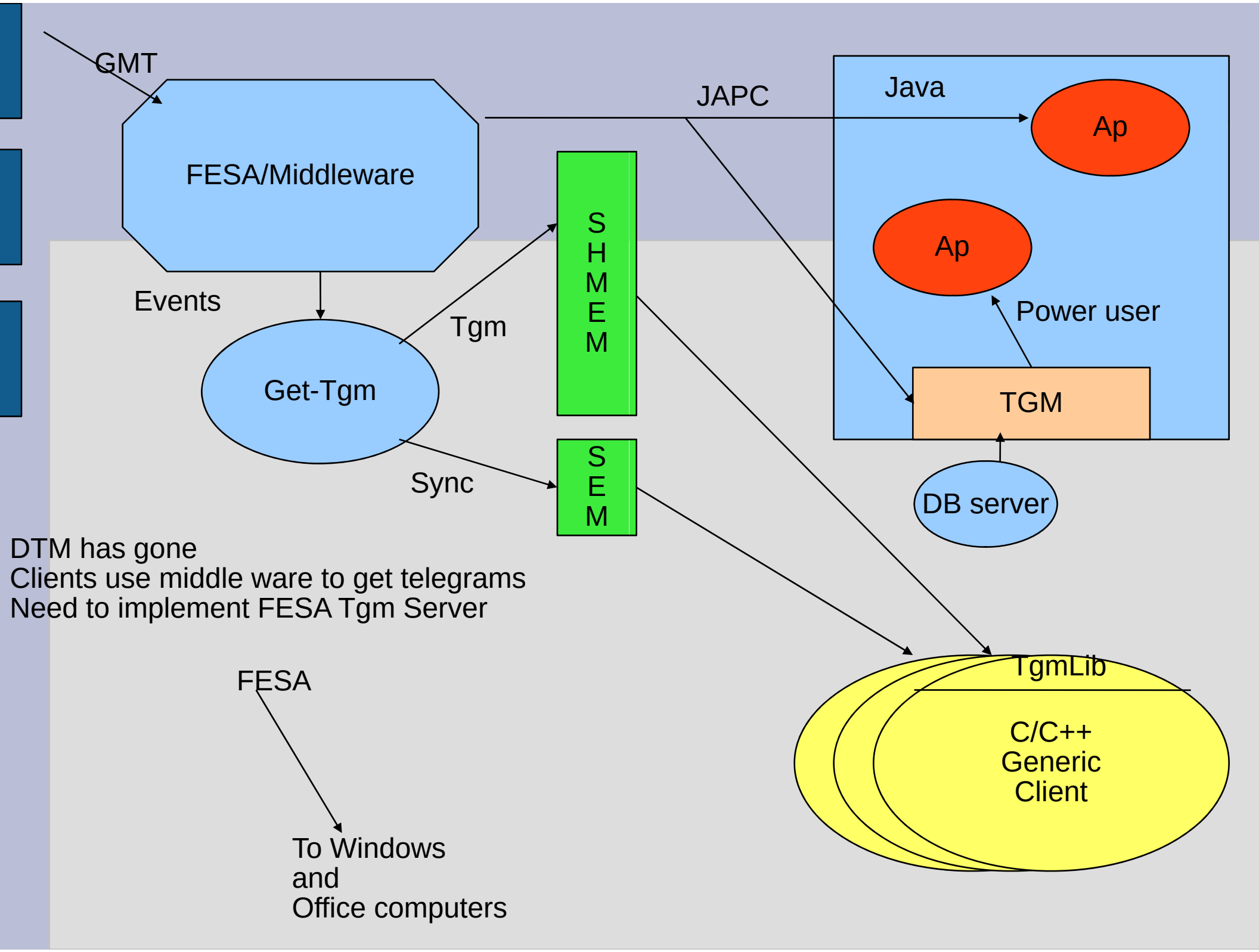
Data Base Cycle Server summing up

- The TGM Java Package functionality is replaced by a server under DM/OP responsibility.
- This server has a cache which is updated when a new Central timing configuration is available. (by an event if VA, or by telegram).
- This server could be extended to cover the VA needs should we decide to go that way.
- Only telegram event handling is left in TGM, and it would go away completely should we decide on VA.
- Nothing is wasted here, no matter what we decide

Quick Fix a stepping stone

- Quick Fix
  - Looks at what we have today and fixes users immediate complaints.
  - This makes use of controls standards and suppresses DTM
  - It does not make any fundamental changes, and keeps a lot of old code running in Java and in the front ends.
  - We would throw away some of this work if we choose VA. (E.g. FESA subscriptions to the telegrams).


GMT
DTM
Events
Get-Tgm
Tgm
Sync
SHMEM
SEM
Java
Ap
Ap
TGM
RMI server
Socket
TgmLib
C/C++
Generic
Client
To Windows
and
Office computers


GMT
FESA/Middleware
JAPC
Java
Ap
Ap
Power user
Events
Tgm
Get-Tgm
SHMEM
SEM
Sync
TGM
DB server
DTM has gone
Clients use middle ware to get telegrams
Need to implement FESA Tgm Server
FESA
To Windows
and
Office computers
TgmLib
C/C++
Generic
Client

Quick Fix

- Is the middleware subscription mechanism adequate and can it handle the large load ?
  - Yes
- Java clients subscribe in standard way to a FESA telegram server.
  - So we need to implement a telegram server
- Legacy clients can still use the Tgm package
  - We add a new back end to the Tgm Java package to call the FESA server.
- C and C++ still supported
  - We add a new back end to get_tgm daemon
- No more DTM .. Yeah !!
- Easy to do, but orthogonal to VA

Questions

- Should we multiplex on payloads in the PS ?
- Should we launch a study on the consequences of VA ?
  - Should we replace USER by VA ?
    - Yes
      - Should we stop sending telegrams ?
      - Should we go for Quick Fix or Big bang ?
      - Should we go for Open CBCM ?
    - No
      - We must implement Quick Fix
      - We must keep telegrams
      - Can we still use VA for Open CBCM ?
        - No
        - How to maintain the existing system ?
        - How to improve it and open it up ?