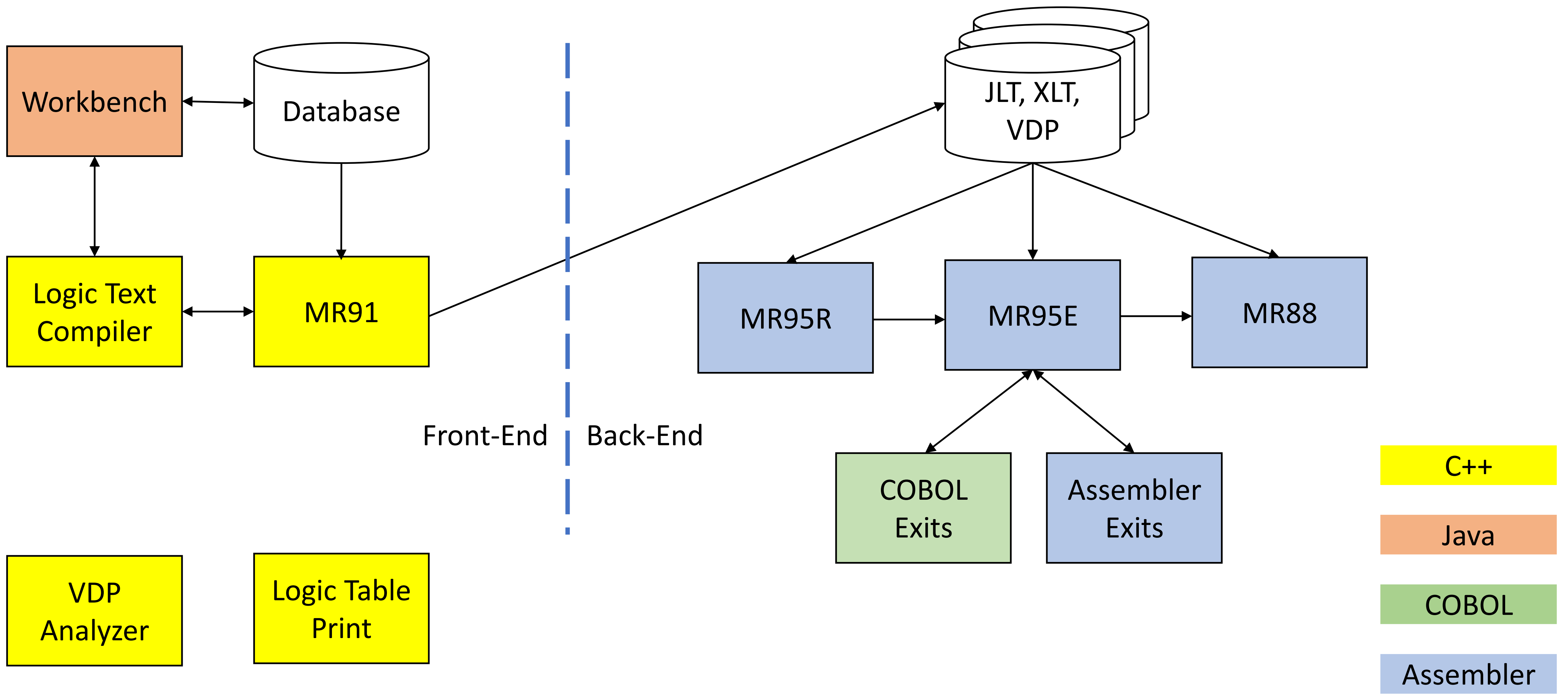
GenevaERS V5 Architectural Options

Assumptions

- The unique strength of GenevaERS is its design, which is optimized for high-volume data transformations:
    - The data model
    - The Logic Table
    - The Extract Engine, which includes:
        - A Logic Table compiler
        - A process which spawns multiple parallel threads to traverse data
        - Multiple data lookup techniques
        - An in-memory data summarization process
- A Logic Table and VDP can currently be derived from views defined in a database, in Workbench XML, or in VDP XML, but GenevaERS may be better served by having its own programming language, perhaps based on Apache Groovy (https://groovy-lang.org/)

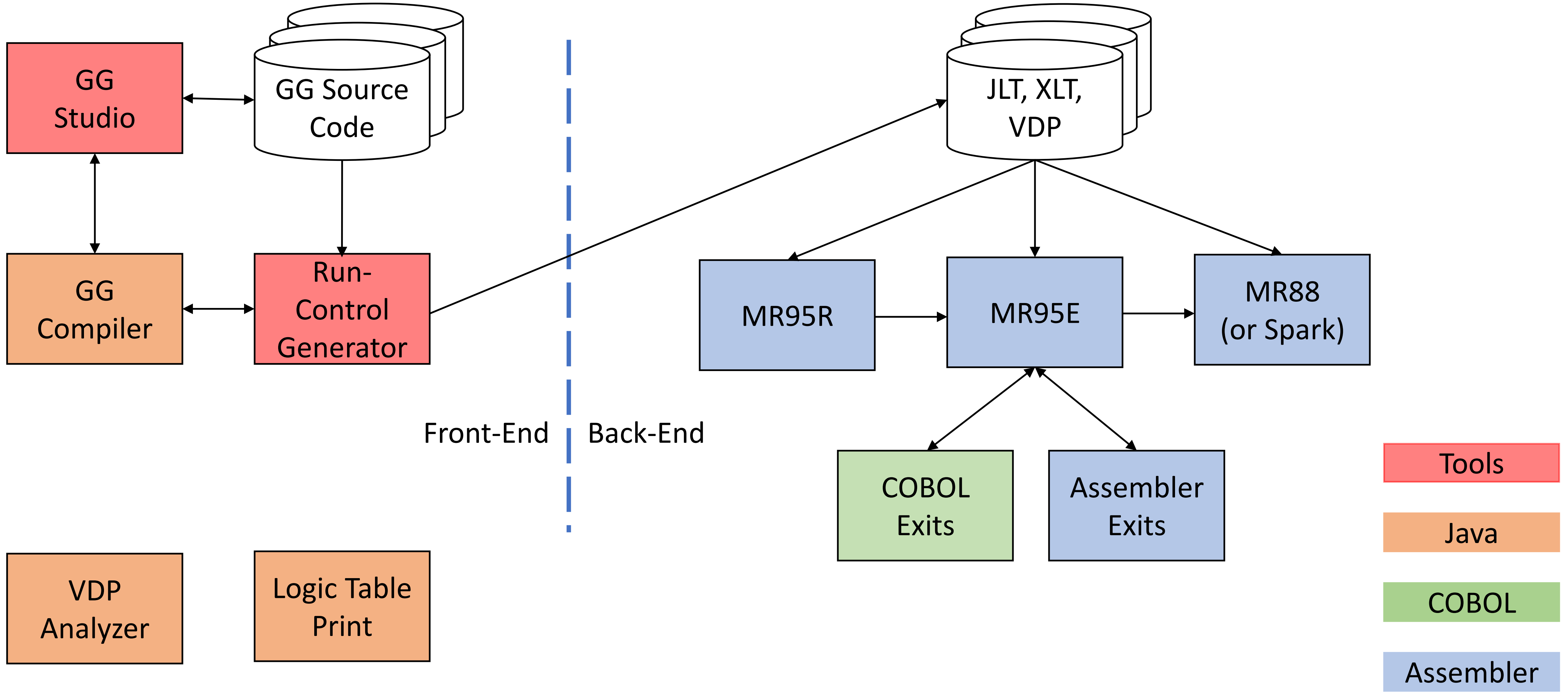
V4 (z/OS) - Current



V4 (z/OS) - Current

- Advantages
  - No changes are needed if we stay with this architecture
  - With the time saved, we could add new features
- Disadvantages
  - Being reliant on a database adds complexity
  - Source code management of GenevaERS objects (views, LRs, etc.) is not supported well

V5 (z/OS) – Use the new Groovy Geneva language



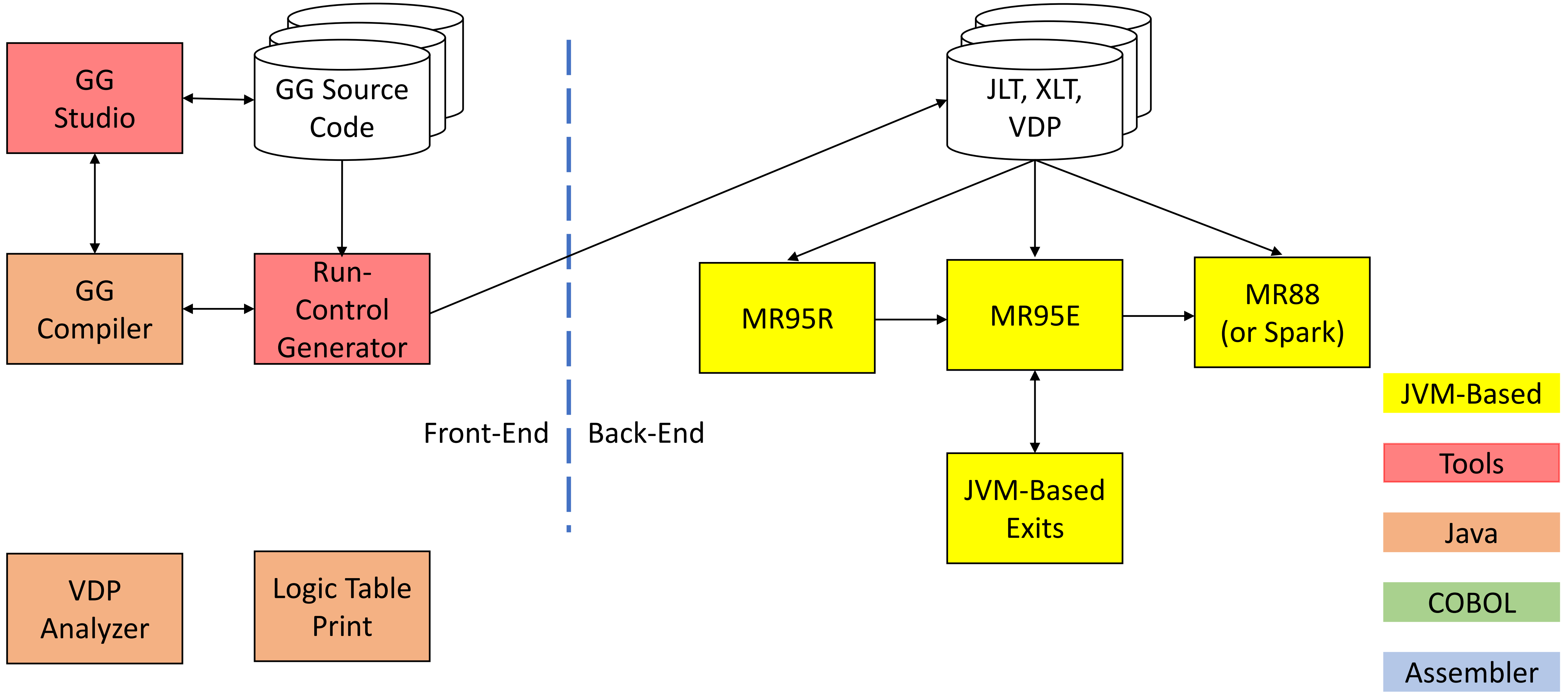
V5 (z/OS) – Use the new Groovy Geneva language

- Advantages
    - Potentially reduces the skill set required to support the product
    - Simpler than the current system
    - Eliminates the complexity caused by the database
    - Lends itself to proper source code management
    - Allows us focus on new features in a more modern development environment by functionally stabilizing the Workbench, the logic text compiler, the database scripts, the stored procedures, the Workbench XML, and the VDP XML
    - New front-end is still compatible with current back-end
        - If a current customer wanted the new front-end, we would perform a one-time conversion of their existing repository to GG files
    - More likely to attract open-source contributors than V4 (z/OS)

V5 (z/OS) – Use the new Groovy Geneva language

- Disadvantages
  - Reduces the skill set required by the support team only after our customers have converted from the current system
  - The functionally stabilized components may be difficult to support if no new development is occurring on them

V5 (JVM) – Make the back-end multi-platform



V5 (JVM) – Make the back-end multi-platform

- Advantages
  - Has all of the advantages of V5 (z/OS) except performance
    - Customers wanting the highest performance can stay on V5 (z/OS)
    - The front-end programs would be the same for either back-end
    - Gives current customers an easy migration path
  - A Logic Table compiler could be developed for any hardware, improving performance
  - Expands the market for GenevaERS
  - Likely to attract open-source contributors
- Disadvantages
  - Has all the disadvantages of V5 (z/OS)
  - Poorer performance than V5 (z/OS)

Alternate configurations

- Alternate user interface with GG compiler and GenevaERS back-end
  - While the GG Studio supports all the features (and complexity) of GenevaERS, a customer may prefer to build a custom interface more appropriate for end users
    - For example, a business rules maintenance facility could provide a subset of GenevaERS functionality and generate source code to be consumed by the GG compiler and processed by the back-end (either Assembler or JVM-based)
- GG Studio with translator and alternate back-end
  - For existing customers wanting to maintain their investment in their GenevaERS views but have transparency into the executable programs, a translator could be developed to convert GG programs to standard Java programs or scripts for another tool, such as Spark