General recommendations:

Material definitions:

- The material definitions can be shared via a class like Em10Materials in TestEm10
- One set of materials can be shared with all examples, the materials should be built preferable via NIST manager, explicitly defined materials can be demonstrated on materials which (for any reasons) cannot be built via NIST manager.

Detector construction:

- Only few geometry setups are used in the examples:
    1. "Container" = World of box shape (TestEm0, 1, 6, 13, 14, 15, 17, 18)
    2. "Ecal" = World of a tube shape (TestEm2, 4)
    3. World with a box absorber optionally with tallies (TestEm5, 7)
    4. World of sphere shape with layers (TestEm12)
    5. And non trivial geometries: TestEm3, 8, 9, 10, 11
- Geometry setups 1 and 2 can be defined by shared classes
- Update geometry is in most examples implemented via reprocessing ConstructVolumes() once again and setting a new world PV to G4RunManager, what may be quite inefficient way if we deal with a realistic geometry. Why G4 standard way via G4RunManager::GeometryHasBeenModified(0 is not used?

Detector construction messengers:

- The commands for setting the detector parameters (materials or dimensions of a selected volumes) can be generalized in a common way
- A prototype for G4UserParameters and G4UserParametersMessenger is in development
- These classes can then simplify DetectorConstruction classes as there will be no need to define data members like fAbsMaterial, fAbsThickness etc. and can be used in PrimaryGeneratorAction to access geometry parameters without a need to pass DetectorConstruction object to this class

Accounted data in user action classes:

- Moving the action classes data members representing accounted data (eg. fEdeposit, fTrackLen etc.) and the methods for their handling in a user defined class (eg. DataManager) can improve code simplicity, readability and re-usability of user action classes:
    - User will find this information in all examples at the same place
    - Stacking, Stepping. Tracking, Event, Run actions can be then shared by most of examples as the data specific actions will be called in a generic way

    - As most of action classes have also their messengers, this can reduce the current number of classes from 18*8=144 to 18*2 + 8 = 44. (Though all examples do not implement all actions, the improvement in maintainability will be still considerable.)
- An example of data manager class and shared user actions was implemented for TestEm1 and new version of this example is provided in the TestEm1.new.tar.gz file

Histograms:

- There should be no need for keeping the histogram data in own HistoManager class; as it is already done in G4AnalysisManager, which should be able to provide all necessary getters. In case something is missing, it is better to add it here than in a user class
- Commands defined in HistoMessenger are now available in Geant4 (tags analysis-V09-05-07, excommon-analysis-V09-05-02)
- The following functions in HistoManager should not be needed:
    - void SetFileName (const G4String& name) { fileName[0] = name;};
    - void save();
    - void SetHisto (G4int,G4int,G4double,G4double,const G4String& unit="none");
    - void FillHisto(G4int id, G4double e, G4double weight = 1.0);
    - void Normalize(G4int id, G4double fac);
    - void PrintHisto (G4int);

Other classes which may be shared:

- StepMax, StepMaxMessenger, SteppingVerbose

Physics Lists:

- Most of SetCuts() methods may be not needed as they are available in G4VUserPhysicsList base class; also setting the cuts values via commands is available in G4 framework

- The commands in messengers for setting cuts can be removed and used G4 command instead:
    - /testem/phys/setGCut
    - /testem/phys/setECut
    - /testem/phys/setPCut
    - /testem/phys/setCuts

- There are many variants of PLs without a sufficient explanation in README

what is in the “local” physics list different from the standard PL or the standard builder in Geant4

- The variants of PhysListEmStandard with added suffices, eg. PhysListEmStandardSS, PhysListEmStandardNR etc. present in more examples vary from an example to another. It could make users life easier, if these named variants are represented by the same (shared) class and the differences are applied by different setting of the class data