Precompound/De-excitation Interfaces

1. **Design**
2. **Interface vs. handling**
3. **Example of mis-using probability**
4. **Changes from Jose Manuel Quesada to Precompound**
5. **Summary**

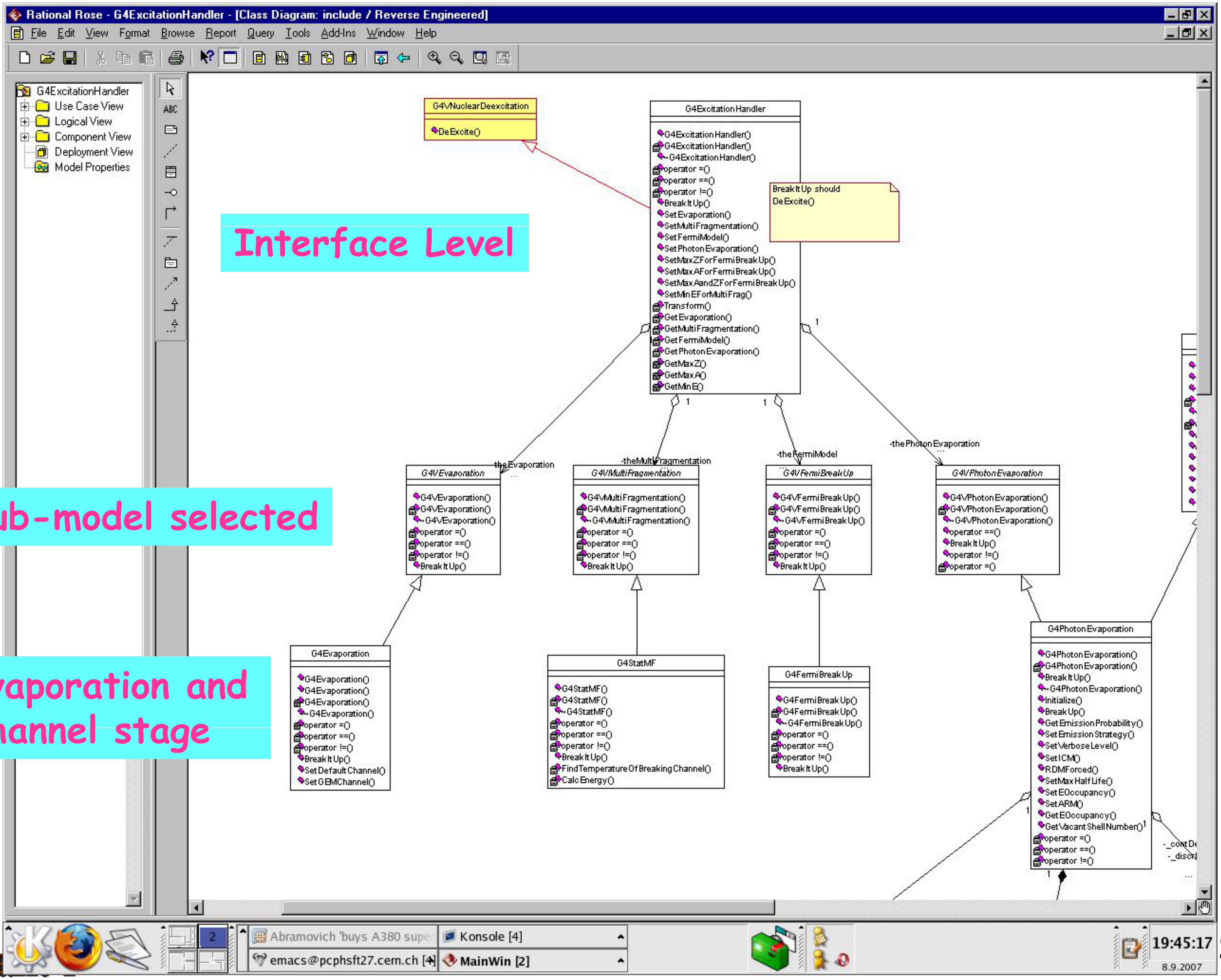
Alex Howard, Gunter Folger CERN

Precompound/De-excitation Interfaces

Geant4 Users Workshop Hebden Bridge 17th September 2007

Design

- After looking at the design for the pre-compound and de-excitation with relevant handling there appears no reason to change it
- It is not a unique solution, but certainly adequate
- Care has to be made about sub-channels and their respective probability (it's competitive)
- It is intrinsic that channels are in competition
- The choice of channel is no longer an interface (at that level) but actually a sub-model
- This is where we were getting confused at the PreCompound workshop
- In choosing evaporation over gamma de-excitation or fission is not simply choosing an interface, the probability has to be handled in a consistent and coherent manner



Interface Level

Sub-model selected

Evaporation and Channel stage

Design (continued)

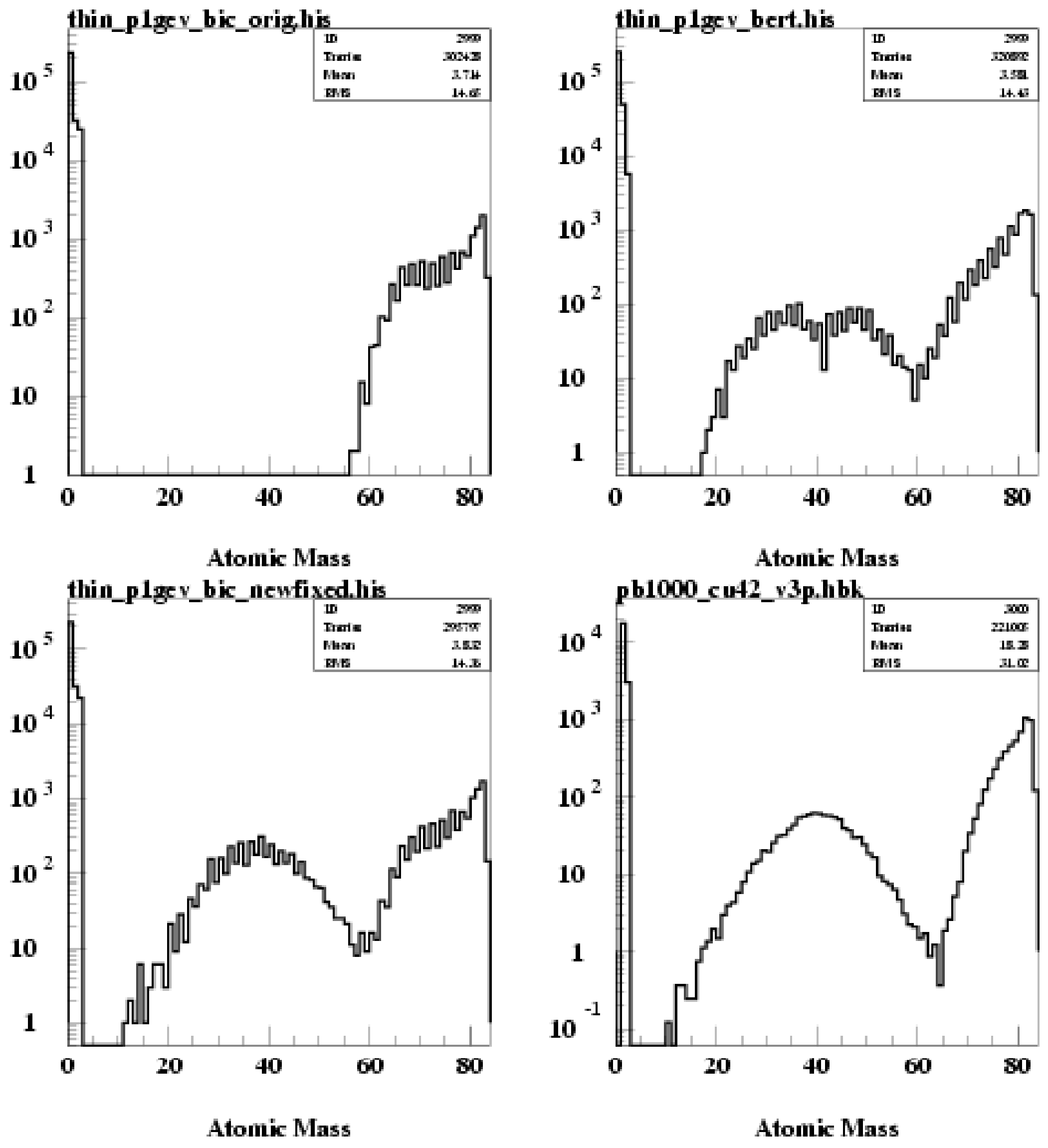
- The interface enters the top-level of the model
- Sub-models are selected on the basis of criteria (target A,Z; excitation energy etc…)
- Probability only enters into the channel selection
- Handler is also required
- Evaporation is dependent on all of the above

Interface to PreCompound (from mini-workshop)

- Main topic was the interface to 'precompound' like models
- Agreement:
  - Use Class (similar) to G4Fragment:
  - ReactionProductVector * res = Propagate(G4Fragment * frag);
  - Also used in interface for deexcitation: G4ReactionProductVector * BreakItUp(const G4Fragment &theInitialState) const;
- Extension of G4Fragment with vector of exciton momenta could be made (no model using this, some associated over-head)

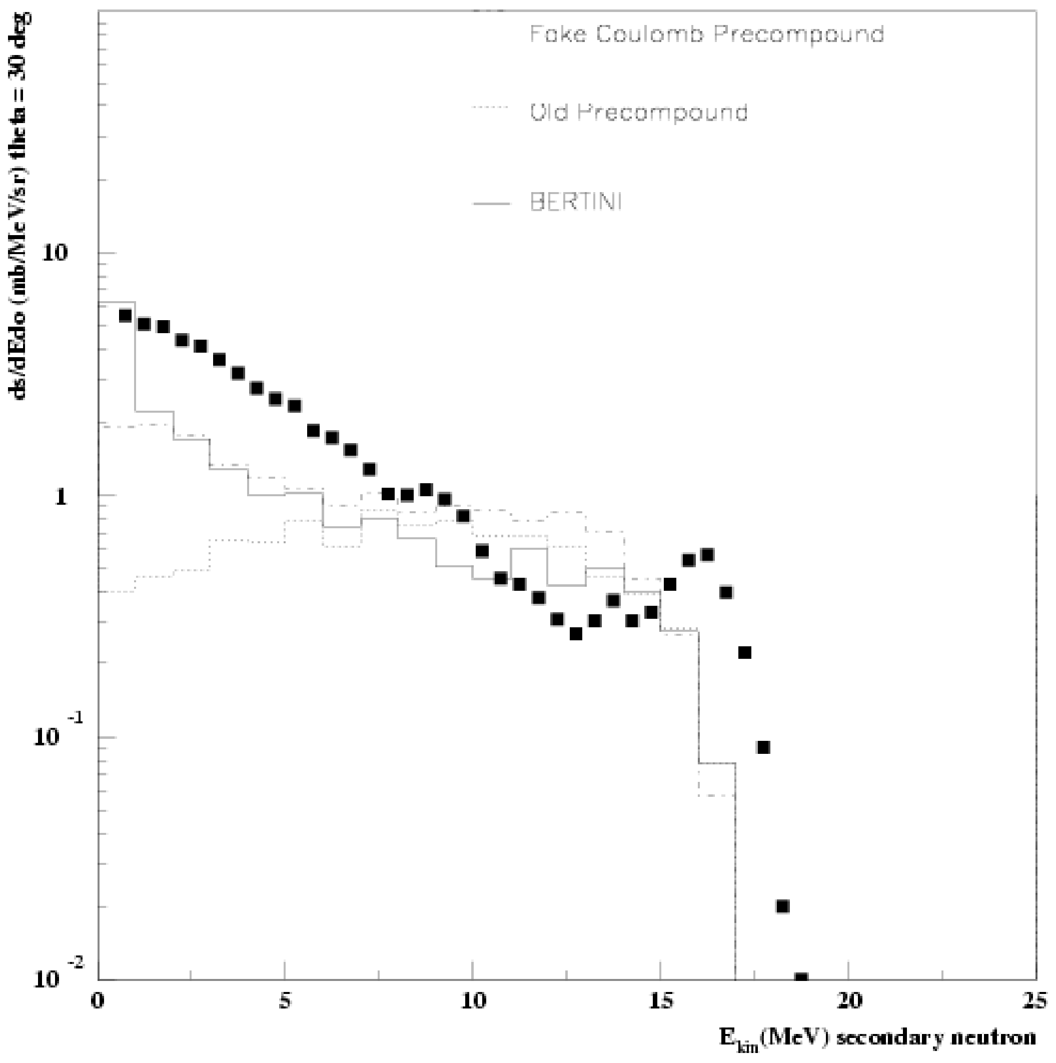
Fission given a chance:

- A bug was found in the fission channel probability which made it negative
- Not only did this suppress fission completely it also produced more gammas due to a reduction in the total evaporation probability



Fake Improvement

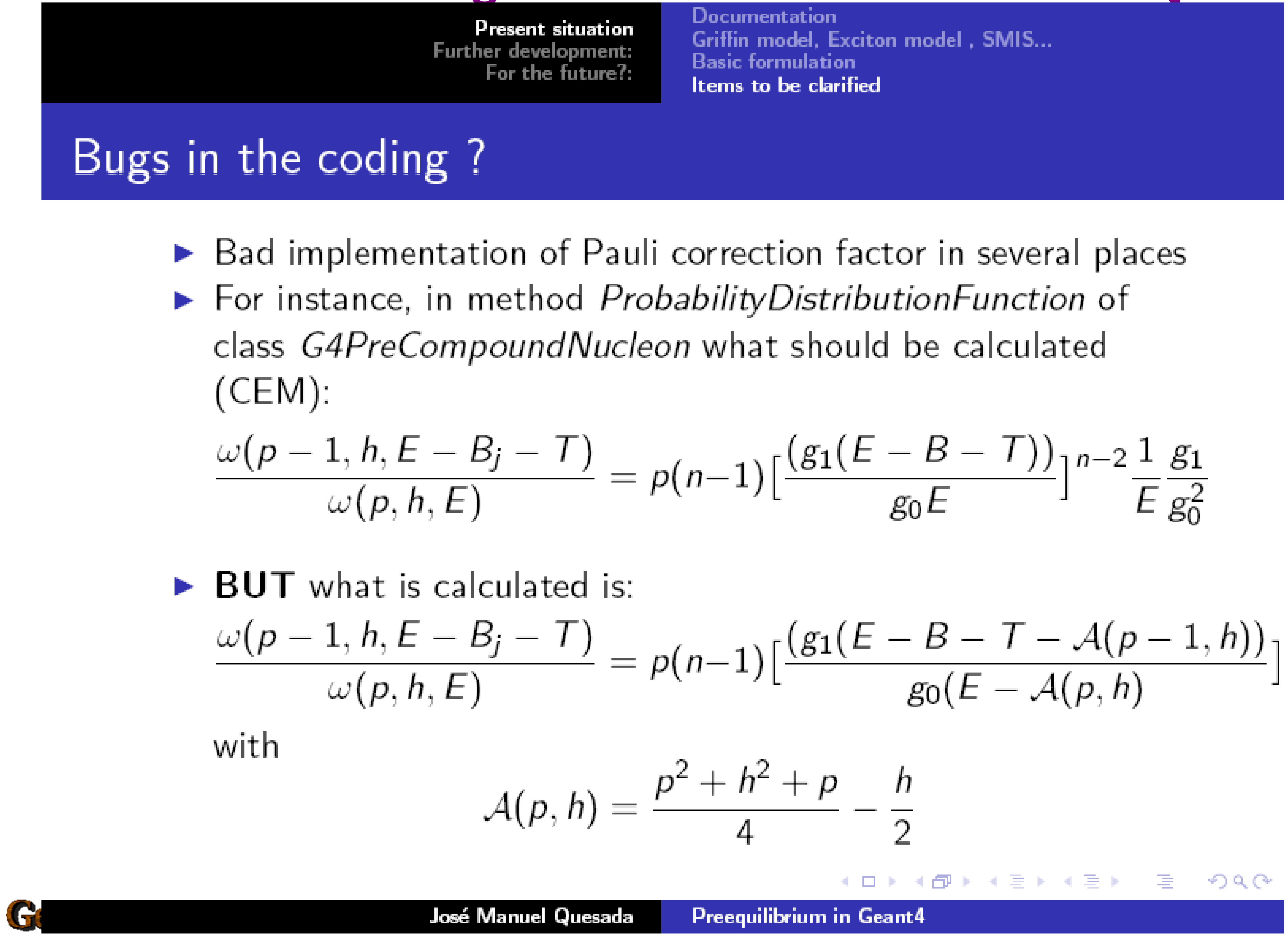
- Double-counting the Coulomb barrier suppressed all charged fragments and correspondingly increased the neutron yield



Bugs in PreCompound/Deexcitation?

- Jose Manuel Quesada (first talk) identified a number of features and bugs in the implementation of PreCompound within Geant4
- Some of these fixes have been implemented with small improvements to the thin target validation
- However, it's not clear we're being completely consistent, particularly with combinatoric/picking factors (R_j)
- Further work needed…

Pauli blocking and other fixes from JMQ:



Bugs in the coding ?

- Bad implementation of Pauli correction factor in several places
- For instance, in method *ProbabilityDistributionFunction* of class *G4PreCompoundNucleon* what should be calculated (CEM):

$$\frac{\omega(p-1,h,E-B_j-T)}{\omega(p,h,E)} = p(n-1)\left[\frac{(g_1(E-B-T))}{g_0 E}\right]^{n-2}\frac{1}{E}\frac{g_1}{g_0^2}$$

- **BUT** what is calculated is:

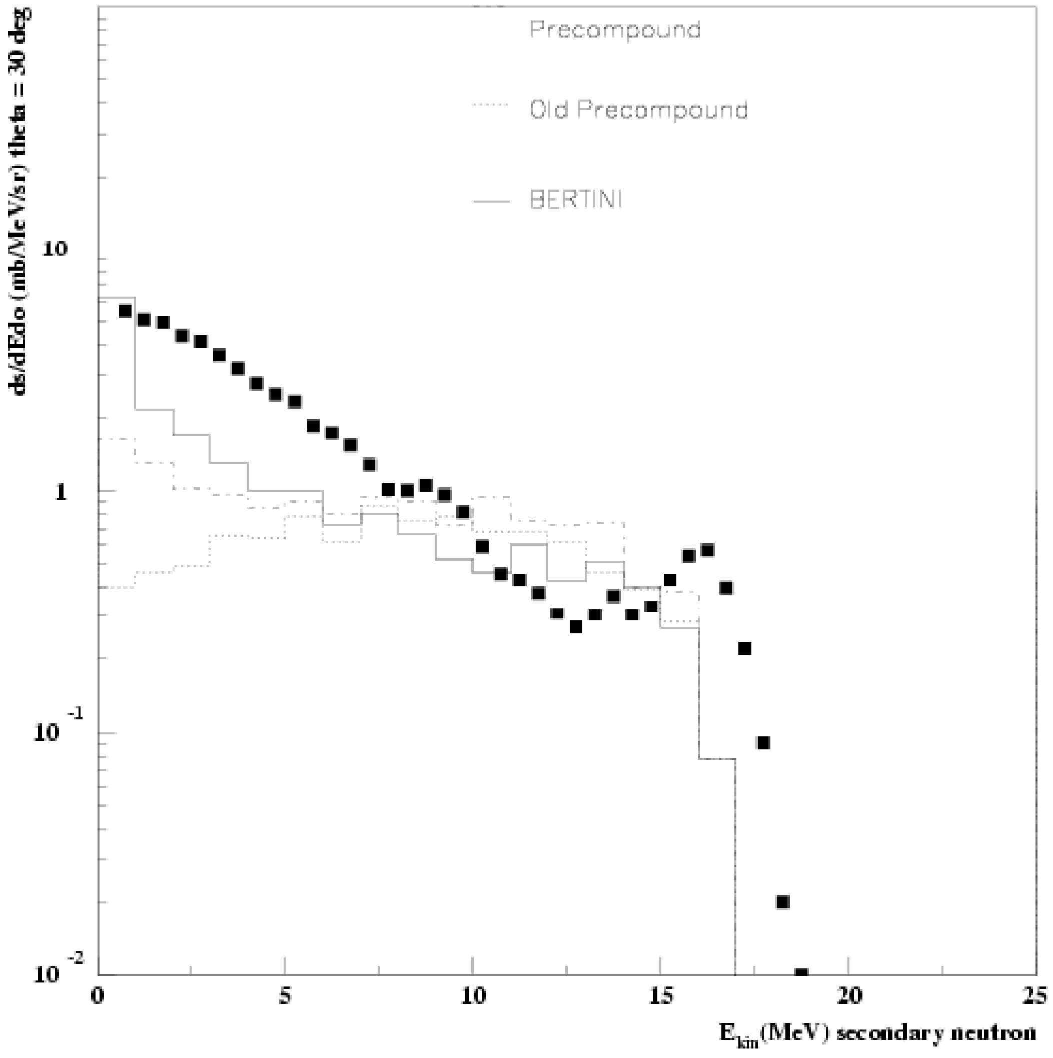
$$\frac{\omega(p-1,h,E-B_j-T)}{\omega(p,h,E)} = p(n-1)\left[\frac{(g_1(E-B-T-\mathcal{A}(p-1,h))}{g_0(E-\mathcal{A}(p,h)}\right]$$

with

$$\mathcal{A}(p,h) = \frac{p^2+h^2+p}{4} - \frac{h}{2}$$

R_j factors/Combinatorics

- With R_j factors included, the heavier fragments (alpha, D, T) are suppressed which increases the low energy yield of neutrons



Summary

- The design of PreCompound/De-excitation seems fundamentally okay
- Although it could be changed it's not clear there's any gain at this stage
- It's important to realise the dependent nature of channel selection, sub-model and evaporation fragment within the de-excitation stage
  - This has to be consistent and coherent
- G4Fragment (in) and ReactionProductVector (out) is sufficient for the models we have in Geant4 at the moment
- In the future an exciton vector may be required to be included as an extension, if new models become available that require lorentz invariance…

- **Spare slides**

Discussion Tuesday 17 July

- Main topic was the interface to 'precompound' like models
- Agreement:
  - Use Class (similar) to G4Fragment:
  - ReactionProductVector * res = Propagate(G4Fragment * frag);
  - Also used in interface for deexcitation: G4ReactionProductVector * BreakItUp(const G4Fragment &theInitialState) const;

G4Fragment

- Ctor to create a fully specified fragment
- Less Set.. than current implementation
- G4Fragment has Get… for
  - Int A,Z mass number and charge (NOT double)
  - Excitation energy; NO Set…
  - Number of excitions, particles, charged particles, holes; note excitions=particles+holes
  - Momentum
  - Secondaries of previous phase (const ReactionProductVector * const)

G4Fragment

- **Ctor:**
- **G4Fragment(const G4int A, const G4int Z, G4int Holes, G4int Particles, G4int Charged, const G4LorentzVector aMomentum, const G4ThreeVector * AngularMomentum, const G4ReactionProductVector * const);**
- **G4Fragment(const G4int A, const G4int Z, G4double excitationEnergy, const G4ThreeeVector * AngularMomentum=0);**
- **G4Fragment(const G4ParticleDefinition *,G4LorentzVector & aMomentum);**

G4Fragment

- **Get Methods**

```
G4int GetA(void) const;
G4int GetZ(void) const;

G4double GetExcitationEnergy(void) const;
const G4LorentzVector GetMomentum(void) const;
const G4ThreeVector GetAngularMomentum(void) const;
            void SetAngularMomentum(const G4ThreeVector value);
G4int GetNumberOfExcitons(void) const;
G4int GetNumberOfHoles(void) const;
G4int GetNumberOfCharged(void) const;
void SetNumberOfCharged(const G4int value);

G4int GetNumberOfParticles(void) const;

G4ParticleDefinition * GetParticleDefinition(void) const;

G4double GetCreationTime(void) const;
      void SetCreationTime(const G4double time);
```

G4Fragment

```
// Some utility methods

  inline G4double GetGroundStateMass(void) const;

  inline G4double GetBindingEnergy(void) const;

#ifdef PRECOMPOUND_TEST
  G4String GetCreatorModel() const { return theCreatorModel; }
  void SetCreatorModel(const G4String & aModel)
  { theCreatorModel = aModel; }
#endif
```