MVC:
In general am urmarit ca patternul MVC implementat de voi sa aiba urmatoarele caracteristici:
1. Repartizarea rolurilor: in special, modelul nu are voie sa contina afisare pe ecran. Aceasta este responsabilitatea View-urilor (-1p daca exista incalcari)
2. Trebuie as exist paternal observer intro model is view-uri, astral incat atunci cand are loc o modificare in model, viewurile sun annotate automat printr-o notificare. (-1p daca nu exist paternal observer intre model is view)
3. Separarea intre model is view ar if trebuit as se supuna principiului dependentelor inverse (DIP), pentru ca scopul urmarit este de a decupla aceste doua componente dpdv al modificarior ce intervin in design. Concret ar fi trebuit sa exise interfete. (- sau -1p in functie de gravitate)
4. Avand in vedere ca M, V si C sunt 3 parti majore ale aplicatiei, era bine de separat aceste componente in 3 packages distince ("-" daca nu s-a facut)

Update:
In general nu am depunctat acolo unde designul era de asa natura incat valorile afisate se recalculau la afisare si deci nu era necesar sa existe neaparat un mecanism de notificare interna. In schimb, am incercat sa tin seama de cum a fost implementata situatia in care se modifica tipul unei celule prin inlocuirea celei vechi cu alta noua.

Facade:
In general am scazut un punct daca fatada nu realiza o diferenta suficient de mare in nivelul de abstractizare al interfetei subsistemului. Concret, ar fi fost bine ca in fatada sa nu mai operam cu clase interne modelului (gen Component sau Expresssion) ci sa aem o interfata in termeni doar de coordonate si de strings introduse de la tastatura de utilizator.