An **intelligent tutoring system** is a computer system that tutors students in some domain of study. For example, in a tutoring system to teach elementary physics, such as mechanics, the system may present the theory and worked-out examples. The system can ask the student questions and it must be able to understand the student's answers, as well as determine the student's knowledge based on what answers were given. This should then affect what is presented and what other questions are asked of the student. The student can ask questions of the system, and so the system should be able to solve problems in the physics domain.

An intelligent tutoring system has the following as inputs:

* Prior knowledge, provided by the agent designer, about the subject matter being taught, teaching strategies, possible errors, and misconceptions of the students.
* Past experience, which the tutoring system has acquired by interacting with students, about what errors students make, how many examples it takes to learn something, and what students forget. This can be information about students in general or about a particular student.
* Preferences about the importance of each topic, the level of achievement of the student that is desired, and costs associated with usability. There are often complex trade-offs among these.
* Observations of a student's test results and observations of the student's interaction (or non-interaction) with the system. Students can also ask questions or provide new examples with which they want help.

The output of the tutoring system is the information presented to the student, tests the students should take, answers to questions, and reports to parents and teachers.

Each dimension is relevant to the tutoring system:

* There should be both a hierarchical decomposition of the agent and a decomposition of the task of teaching. Students should be taught the basic skills before they can be taught higher-level concepts. The tutoring system has high-level teaching strategies, but, at a much lower level, it must design the details of concrete examples and specific questions for a test.
* A tutoring system may be able to reason in terms of the state of the student. However, it is more realistic to have multiple features for the student and the subject domain. A physics tutor may be able to reason in terms of features that are known at design time if the examples are fixed and it is only reasoning about one student. For more complicated cases, the tutoring system should refer to individuals and relations. If the tutoring system or the student can create examples with multiple individuals, the system may not know the features at design time and will have to reason in terms of individuals and relations.
* In terms of planning horizon, for the duration of a test, it may be reasonable to assume that the domain is static and that the student does not learn while taking a test. For some subtasks, a finite horizon may be appropriate. For example, there may be a teach, test, reteach sequence. For other cases, there may be an indefinite horizon where the system may not know at design time how many steps it will take until the student has mastered some concept. It may also be possible to model teaching as an ongoing process of learning and testing with appropriate breaks, with no expectation of the system finishing.
* Uncertainty will have to play a large role. The system cannot directly observe the knowledge of the student. All it has is some sensing input, based on questions the student asks or does not ask, and test results. The system will not know for certain the effect of a particular teaching episode.
* Although it may be possible to have a simple goal such as to teach some particular concept, it is more likely that complex preferences must be taken into account. One reason is that, with uncertainty, there may be no way to guarantee that the student knows the concept being taught; any method that tries to maximize the probability that the student knows a concept will be very annoying, because it will continue to repeatedly teach and test if there is a slight chance that the student's errors are due to misunderstanding as opposed to fatigue or boredom. More complex preferences would enable a trade-off among fully teaching a concept, boring the student, the time taken, and the amount of retesting. The user may also have a preference for a teaching style that should be taken into account.
* It may be appropriate to treat this as a single-agent problem. However, the teacher, the student, and the parent may all have different preferences that must be taken into account. Each of these agents may act strategically by not telling the truth.
* We would expect the system to be able to learn about what teaching strategies work, how well some questions work at testing concepts, and what common mistakes students make. It could learn general knowledge, or knowledge particular to a topic (e.g., learning about what strategies work for teaching mechanics) or knowledge about a particular student, such as learning what works for Sam.
* One could imagine that choosing the most appropriate material to present would take a lot of computation time. However, the student must be responded to in a timely fashion. Bounded rationality would play a part in ensuring that the system does not compute for a long time while the student is waiting.